Visual CUT

View, Export, Burst, Email, and Schedule Crystal Reports

www.MilletSoftware.com

Version 6.9001
December 2015

by
Millet Software
5275 Rome Ct.
Erie, PA 16509-3951
ido@MilletSoftware.com
(814) 825-6009

Disclaimer: This software is provided as-is by Millet Software without assuming any responsibility for harm to computer systems, software, or data with which these files are used.

The following individuals were instrumental in providing suggestions and support leading to enhancements: Ken Hamady, Dave Clutter (Reed Manufacturing), Nancy Peterson (JobMark.com), David Parker (VendorsInfo.com), Jim Woodin (Diamond V Mills), Dave Hawkins (Orbitz), José Alsina (Innovatec), Jorge Vazquez (Merck), Mike Marsten (UCSF), Helen Hiebert (Independent Consultant), Dave Neubauer (Bekaert Specialty Films), Nathaniel Wetherbee (PFPC), Ben Pomeranz (WorldCom), Shawn Wright (Shawnigan Lake School), Bernard Paes (Flag Choice Hotels), Larry Bodie (Kelsan), Abbas Rostami (Cubic Corporation), Rick Scero (SITEL), Jon Palmbak (LeBoeuf, Lamb, Greene & MacRae), Bill Arruda (Pragmeta), Larry Bates (Syscon), Lori Fraticelli (K. Hovnanian Enterprises), Vik Mohindra (Spryer Soft), Blaise Masse (University of New Hampshire), Greg Davis (Act Solutions), Don Gilsdorf (Gain Focus Technologies), Tim Dunevant (The Matworks), Chris Kell (Washington State University), Ken Rickard (EMU), Jonathan Washam (Independent Portfolio Consultants), Greg Scharer (Upstate), Sharon Mone (Fujitsu), Tom Cook (Grotenhuis), David Leland (Johnson Corporation), Ron Ruth (TIB Bank of the Keys), Jim Stickley (TraceSecurity), Dave Atkins (The Matchett Group), Egil Stenberg (Flextronics), Peter Sclafani (Daiwa Securities America), Kelly Serge (DataSafe), Walid El Khazen (Wolfram Research), Bill Loucks (Westminster College), David Hopaluk (Here2Help Solutions), Frank Schwarz (Orbitz), Dixie Folzenlogen, Denise Hirata (Retirement.org), Sateesh Annangi (Keenan), Todd Erickson (Technology Navigator), Peter Enman (RCMP), Gene Juhos (Sysmatics), Adam Peter Butt (APBReports), Dwight Wyse (RecSoft), Mario Blasius (Globetrotter Corporate Travel), Angelo Claustro (Kelso-Burnett), Steve Jones (Kokatat), Mark Schultze (Lamons), Wim Rippen (Petrovlis-H olland), Paul Contreras (R.B. Zack & Associates), Brian Orr (U.S. Sterling Capital), Tim Cortis (Wayne J. Griffin Electric), Mick McCann (Worldmark International Ltd.), Praveen Rao (Brighthouse Networks), Jon Leeds (Carolina Software), Elliott R Pickar (Continuum Health Partners), Erwin Ponce (ELP Aviation), Terry Thurgood (Foresight Technology), Jeff Schwartz (Bell Nursery USA), Nitzan Neeman (SeniorCare EMS)…

INTRODUCTION & KEY BENEFITS **10**
INSTALL / REMOVE **12**
Selecting a Version: 8.5, 9, or 11 (XI R2) 12
STEP 1: SELECT REPORT **14**
STEP 2: PREVIEW **16**
SPEEDING UP REPORT PREVIEWS 17
CHANGING PROCESSING OPTIONS WITHOUT PREVIEWING 17
CHANGING LOGIN, REPORT PATHS & OTHER SETTINGS WITHOUT PREVIEWING 17
Find & Replace Report Paths and other Saved Settings 18
Disabling Find & Replace Categories 18
SAVE AND REUSE NAMED PARAMETER SETS 19
STEP 3: EXPORT/BURST/EMAIL… **21**
GROUP VALUES AREA 22
FIELDS & FORMULAS AREA 23
EXPORTING/BURSTING/PRINTING OPTIONS AREA 24
Exporting to Multiple Files/Formats In a Single Pass 24
Replacing Illegal Characters in Dynamic File Names 25
EMAIL OPTIONS AREA 26
Combining Static & Dynamic Content 26
Specifying SMTP Server (optional) 27
Embedding Report as Image in Email Message Body 28
Embedding Pivot Tables in Email Message Body 29
Embedding HTML Export in Email Message Body 30
Embedding TEXT Export in Email Message Body 31
Embedding File(s) Content in Email Message Body 31
Sending Message Text as HTML 32
Integrated HTML Editor 33
Embedding Images inside the HTML email body (old email engine – no longer available) 33
Embedding Images inside the HTML email body (2009 email engine) 33
Dynamic Tables inside HTML Email Messages 34
Applying Alternating Row Color 37
Embedding Hyperlinks to Reports/Files inside HTML Email Messages 38
Using Cascading Style Sheets (CSS) in HTML messages 39
Attaching Multiple Files 40
Attaching Optional Files 40
Specifying Multiple (Simple/Composite) Email Addresses 41
Specifying Email Distribution Lists in Text Files 41
Specifying Email Distribution Lists in SQL Queries 42
Specifying Bursting email Addresses in a Formula 43
Specifying an Email Reconnect Option for Email Bursting 43
Specifying a Different Character Set 43
Skipping Emails (VC_Skip_Email) 43
NEW EMAIL ENGINE 44
Key Features 44

QUEUING EMAILS & THE SMTPQ SERVICE 46
Monitoring Email Queuing 46
smtpQ Administration 47
Naming Scheme of .eml Files 48
smtpQ Service Failure Action Properties 49
Slowing Down Outgoing Emails 50
GENERATING TWITTER OR SMS MESSAGES VIA EMAIL 51
'START PROCESS' BUTTON AND PROGRESS WINDOW 52
SCHEDULING AREA 53
Scheduling String 53
Arguments 53
Scheduled Printing 53
SCHEDULING 54
PLACE THE COMMAND LINE IN A BATCH (.BAT) FILE 54
Command Line Wizard (Parameters & Skip Zero Records) 55
PROCESSING OF MULTIPLE REPORTS IN A SINGLE BATCH (.BAT) FILE 56
CALL THE BATCH FILE FROM WINDOWS TASK SCHEDULER 56
SCHEDULING ISSUES 61
Mapped → UNC File Paths 61
Windows 7/8 Task Scheduler 61
Minimize or Hide the Command Window 61
CONTROL PROCESSING OPTIONS WHEN ENCOUNTERING ZERO RECORDS 62
E-MAILING ALERT MESSAGES AND EXCEPTION REPORTS 62
USING COMMAND LINE ARGUMENTS 63
ARGUMENTS TO SPECIFY PARAMETER VALUES 63
Range and Multi-Value Parameters 64
Request User Input for Certain Parameters 64
Null Values 64
Date Constants 65
Custom Calendars 66
Adjusting Data Constants for Day of Week 67
Number Constants 68
Reports with More Than 8 Parameters (restriction removed, 6.2002) 69
Using Parm8 to Specify all Extra Parameter Values 69
Using PowerShell to Set Relative Date Parameters and Call Visual CUT 70
Using Command Line Argument to Set Formula Expressions 71
ARGUMENTS TO SPECIFY PRINTER DESTINATION 72
Printing to Multiple Printers 72
Using a Text File to Specify Multiple Printers 72
ARGUMENTS TO SPECIFY PRINTER BURSTING DESTINATION 73
SPECIFYING NUMBER OF COPIES 73
SETTING CUSTOM TEXT FOR EACH PRINT COPY 73
ARGUMENTS TO SPECIFY USER ID & PASSWORD 74

SETTING ENCRYPTED PASSWORD ENTRIES 74
DATABASE CHOICE FUNCTIONALITY (COMMAND LINE / GUI) 76
Selecting an Alternative ODBC Data Source 76
Overriding the Database Specified in the Report or ODBC DSN 78
Overriding ODBC Table Location (.CSV Files as Data Source) 78
Overriding the Server in Native Oracle Connection 79
Selecting an Alternative SQL Server – OLE DB Data Source 79
Changing Folder Location for Access/Excel/Pervasive/ACT! Files 80
ARGUMENTS TO SPECIFY EXPORT FORMAT 81
ARGUMENT TO SPECIFY EMAIL PRIORITY 82
ARGUMENT TO SPECIFY EMAIL HEADERS 82
ARGUMENTS TO SPECIFY EMAIL OPTIONS 83
ARGUMENTS TO PROCESS REPORTS WITH NO SETTINGS 83
CALLING VISUAL CUT FROM ANOTHER APPLICATION 84
SPECIFYING ARGUMENTS FROM THE GUI 85
REFERRING TO SAVED ENCRYPTED PASSWORDS 86
OTHER OPTIONS AND FEATURES 87
DELEGATING EXPORTING/BURSTING TO DATALINK VIEWER 2011 87
TRIGGERING A BATCH FILE WITH DYNAMIC CONTENT BEFORE/AFTER EXPORT 88
Triggering Reports Based on Database, File, and Email Events 89
PRINTING 89
Bursting to Printer 89
Interweaving Burst Printouts From Multiple Reports 90
Printer Job Name Functionality 91
ZIP AND PASSWORD PROTECT FILES 92
LOAD INI VALUES INTO PARAMETERS 93
Securing Reports against Unauthorized Use 93
FTP/SFTP 94
Exporting to an FTP Server (old approach) 94
Uploading to an FTP Server (new approach) 95
Uploading to an SFTP Server 96
Downloading from an FTP Server 97
EXTRACT FILES STORED IN DATABASE 98
FILE LOCATION FUNCTIONALITY 99
Automatic Handling of Write-Protected Application Folders 99
DIRECTING THE VISUAL CUT DATABASE TO ANOTHER DBMS 100
UPDATING DATALINK_VIEWER.INI VIA A DELTA FILE 101
RESTRICTING USER ACTIONS 102
INTEGRATED INTERACTIVE AUTHENTICATION 103
Integrated Authentication ("Remember Me") 103
Shared Machine Authentication 104
AUTO-REFRESHING WEB DASHBOARDS 105

WEB DASHBOARD EXPERT 105
IFRAME APPROACH 106
PREVENTING FILE LOCKING 106
CASE STUDY 106
HYPERLINKS 107
TOOLTIPS 107
GENERATING HTML VIA EMAIL MESSAGE BODY 107
CLEANING PNG FILE REFERENCES 108
RESTRICTING ACCESS TO WEB DASHBOARDS 108
PDF FUNCTIONALITY 109
CREATING BOOKMARKS (GROUP TREE) IN EXPORTED PDF FILES [OLD APPROACH] 109
Controlling PDF Bookmark Colors (& Text) 110
Controlling How Many Bookmark Levels Are Initially Expanded 111
Guarding Against Null Bookmark Values 111
Generating PDF Bookmarks from within Subreports 112
ADDING BOOKMARKS USING CRYSTAL FORMULAS AS TAGS [NEW APPROACH] 113
Setting Up a Crystal Report with pdf formula tags 113
ADDING A TABLE OF CONTENT TO A PDF FILE 115
Advanced Table of Content Options 116
ADDING PAGE NUMBERS TO A PDF FILE 117
ADDING WEB/FILE/EMAIL HOTSPOT TO A PDF FILE 118
ADDING AN IMAGES WITH AN OPTIONAL HOTSPOT TO A PDF FILE 119
ADDING LINKS/IMAGES TO FILES/PAGES USING CRYSTAL FORMULAS AS TAGS 120
Setting Up a Crystal Report with pdf field tags 120
Formula Example from the Sample Report 121
EMBEDDING FILES AS ATTACHMENTS INSIDE A PDF FILE 122
ADDING LINKS AND EMBEDDED FILES USING CRYSTAL FORMULAS AS TAGS 123
Setting Up a Crystal Report with Link_Tag Formulas 123
Formula Example from the Sample Report 124
ADDING LINKS TO FILES BY DETECTING FILE REFERENCES IN PDF TEXT 126
Detecting Additional File References Using Wild Card Tokens 127
ADDING DIGITAL SIGNATURE TO A PDF FILE 128
ENCRYPTING & PROTECTING A PDF FILE 129
MERGING PDF FILES 130
Dynamic File Names 131
Using a Text File to Specify Files for Merging 132
Using Wild Cards to Specify Files for Merging 132
Controlling Merged Bookmark Colors 132
Specifying Bookmarks when Merging PDF Files 133
Using the Merged File Names to Generate Bookmarks 134
Using the Merged File Names to Generate Multi-Level Bookmarks 135
MERGING 1-PAGE PDF FILES INTO LAYERS IN A SINGLE PDF FILE 136

Dynamic File Names 136
Using a Text File to Specify Files for Merging 137
Controlling Layer Name & Visibility 137
PRINTING PDF FILES 138
PRINTING PDF FILES TO MULTIPLE PRINTER TRAYS 139
Using PDF_Print_Split 139
Using PDF_Print_Split_Tag 140
CONTROLLING PRINT QUALITY/SPEED 141
ADDING BACK-PAGES 142
Using PDF_Insert_BackPage 142
SAVING PDF FILES TO IMAGE FILES 143
ADDING FORM FIELDS & SUBMIT BUTTONS TO PDF FILES 144
Sample PDF File with Form Fields & Submit Buttons 144
Sample Crystal Report with Formulas as Form Field Tags 145
Setting Up a Crystal Report with pdf field tags 145
Formula Example from the Sample Report 145
Populating Form Field Text or Description via ODBC Query 147
FILLING PDF FORMS 148
Setting Up a Crystal Report to Act as a PDF Form Filler 148
Setting Up the Report in Visual CUT 148
SETTING PDF DOCUMENT PROPERTIES AFTER EXPORT (AUTOMATIC) 149
FLATTEN PDF FILES 150
SETTING PDF DOCUMENT PROPERTIES 151
BUILDING AN INDEX PDF DOCUMENTS 152
ADDING AN INDEX FILE TO A PDF DOCUMENT 153
ADDING MULTIMEDIA TO PDF DOCUMENT 154
IMPORTING MULTI-PAGE TIFF FILES INTO PDF FILES 155
IMPORTING IMAGE FILES AS NEW PAGES (TAG APPROACH) 156
SPLITTING PDF FILES 157
Splitting By Bookmarks 157
Splitting By Embedded Tags 158
Splitting and Protecting PDF Files By Embedded Tags 159
COMPRESS PDF FILES 161
LINEARIZE (WEB-ENABLE) PDF FILES 162
EXPORT TO PDF VIA MS WORD 163
EXCEL FUNCTIONALITY 164
EXPORTING TO EXCEL 2007 (.XLSX) FILES 164
COMBINING EXCEL BURSTING EXPORTS INTO A SINGLE MULTI-TAB SPREADSHEET 165
ADDING EXCEL EXPORTS AS TABS IN EXISTING/NEW SPREADSHEETS (BRIEFING BOOKS) 166
Appending or Replacing Data for Existing Tabs 166
CONTROLLING EXCEL TAB COLORS 167

SETTING UP PRINT PROPERTIES FOR EXCEL WORKBOOKS 168
AUTO FILTER & FREEZE PANES IN EXCEL EXPORTS 169
COLUMN AUTO FIT IN EXCEL EXPORTS 169
PASSWORD PROTECTING EXCEL WORKBOOKS 170
PROTECTING EXCEL WORKSHEETS AGAINST USER VIEWING/EDITING 171
INSERTING FILE EXPORTS INTO EXCEL TEMPLATES 172
Sample Input & Output 173
INSERTING CRYSTAL VALUES INTO EXCEL TEMPLATES 174
Specifying Named Ranges in the Excel Template and Matching Formulas in Crystal 174
Crystal Formula Content for Multi-cell and multi-row Named Ranges 175
Populating Multiple Named Ranges 176
Before & After Images 177
TRANSFERRING TABS TO ANOTHER WORKBOOK 178
SPLITTING EXCEL WORKBOOKS 179
GENERATING EXCEL PIVOT TABLES 180
REFRESHING DATA (QUERIES, PIVOT TABLES) FROM ALL CONNECTIONS 183
RUNNING AN EXCEL MACRO 184
REPLACING CONTENT IN EXCEL FILES 185
Exporting Formula Expressions to Excel, and Activating Them 185
CONVERT XLS FILES TO XLSX (AND MERGING SHEETS) 186
CONVERT EXCEL FILES TO PDF 187
CONVERT EXCEL FILES TO HTML 188
CONVERT EXCEL FILES TO CSV 189
MS WORD FUNCTIONALITY 190
PROTECTING MS WORD FILES 190
INSERTING CRYSTAL VALUES INTO MS WORD DOCUMENTS 191
Specifying Field/Formula References (tags) in the Word document 191
Populating Word Tables with Crystal Formula Data 193
REPLACE FORMATTING IN MS WORD FILES 195
PRINTING MS WORD FILES 196
PRINTING MS WORD FILES WITH WATERMARKS 197
CONVERT WORD FILES TO PDF 198
TEXT FUNCTIONALITY 199
SPLITTING TEXT FILES BY EMBEDDED TAGS 199
MERGING TEXT FILES 200
REPLACING CONTENT IN TEXT FILES 201
REMOVING BLANK OR SHORT LINES IN TEXT FILES 202
REMOVING GUIDS FROM PNG FILES REFERENCED IN HTML EXPORTS 203
REPLACING CONTENT IN TEXT/HTML FILES – TOKEN APPROACH 204

INSERTING BASE64-ENCODED FILES INSIDE TEXT/XML 206
ODBC EXPORT FUNCTIONALITY 207
Append Functionality 207
Replace Functionality 207
CAPTURING & PROCESSING INCOMING EMAILS 208
USE SCENARIOS 208
TRIGGERING EMAIL CAPTURE 211
[EMAIL_GET] INI SECTION 211
EMAIL GET DIRECTIVE SECTIONS 212
CAPTURING EMAILS 213
Phase 1: Header Download & Filtering 213
Phase 2: Targeted Download & Database Capture 214
Monitoring Results of the Process 214
EMAIL CAPTURE TABLE STRUCTURE 215
MS Access Table Structure 215
SQL Server Table Structure 216
Oracle Table Structure 217
MONITORING VISUAL CUT PROCESSING 218
LOGGING AND MONITORING VISUAL CUT 218
'Log Email Activity' option 218
Silent Unattended Failure Option 219
Silent Attended Failure Option 220
Avoiding Duplicate Processing 221
Avoiding Too Many Active Visual CUT Instances (Queuing) 222
JOB STATUS FUNCTIONALITY 223
FAILURE ALERTS VIA EMAIL 224
RECORD PROCESSING TO AN ODBC DATABASE 225
MS Access Database Sample 225
SQL Server Instructions 226
Process Logging Settings in Options Dialog 227
UPDATE A DATABASE AFTER SUCCESS (AFTER_SUCCESS_SQL) 228
SQL Server Example 1 229
SQL Server Example 2 229
UPDATE A DATABASE BEFORE REPORT RUNS (BEFORE_REPORT_RUN_SQL) 230
CALL A WEB SERVICE AFTER SUCCESS (AFTER_SUCCESS_HTTP) 231
Sending SMS Messages 231
TRIGGER DYNAMIC BATCH FILE AFTER SUCCESS (AFTER_SUCCESS_BATCH) 232
Dynamic References to Fields/Formulas within the Batch File 232
Logging to a text file 232
TABLE OF COMMAND LINE ARGUMENTS 234
UPDATE HISTORY 240
VERSION 6.9002: ENTERED TESTING DECEMBER 13, 2015 240

Fixes 240
VERSION 6.9001: RELEASED NOVEMBER 21, 2015 241
Key Features 241
Excel Features 242
PDF Features 243
Email Features 243
Other Features 244
Fixes 244
VERSION 6.8001: RELEASED NOVEMBER 16, 2014 245
Web Dashboard and FTP/SFTP Features 245
PDF Features 245
Excel Features 246
Email Features 246
Other Features 247
Fixes 249
VERSION 6.7001: RELEASED DECEMBER 01, 2013 250
Email Features 250
PDF Features 251
Excel Features 251
Web Dashboard & FTP/SFTP Features 251
User Interface Enhancements 252
Command Line Arguments 252
Fixes 253
Other 253
VERSION 6.6001: RELEASED DECEMBER 31, 2012 254
VERSION 6.5001: RELEASED JANUARY 26, 2012 259
VERSION 6.4001: RELEASED JUNE 01, 2011 264
VERSION 6.3001: RELEASED JULY 02, 2010 270
New PDF Features 273
VERSION 6.2001: RELEASED NOVEMBER 28, 2009 275
VERSION 6.1001: RELEASED MARCH 10, 2009 279
VERSION 6.0000: RELEASED 11/23/2008 282
KNOWN ISSUES AND LIMITATION 324

Introduction & Key Benefits

Visual CUT lets you **schedule periodic and exception reports** for **exporting, bursting, e-mailing, and printing.** E-mail messages, export file names and folders, number of copies, and many other **options can incorporate dynamic content from fields/formulas in the report via a drag & drop user interface.** The main benefits of ***Visual CUT*** include:

1. **E-mail** a dynamic-content message for the report as a whole or for each Group Level-1 via the widely supported **SMTP** standard. A variety of options are supported including **queued, archived, secured** (NTLM, CRAM-MD5, SSL, STARTTLS)**, encrypted, and signed messages**. Message bodies and options (subject, From, To, cc, bcc, priority, attachments) can **incorporate dynamic content from fields/formulas in the report** (via a drag & drop user interface) and can be **formatted as HTML**.

2. **Export** the report as a whole or **Burst** each Group Level-1 to a variety of file formats, save the results to dynamically named disk files/folders and include them as attachments to email messages.

3. **Schedule** exporting, bursting, emailing, and printing of reports based on parameters and options you specify (and save) during an interactive session.

4. **Control and Invoke** processing of Crystal reports using command line arguments. A full command line interface allows calls from batch files, schedulers (including the free Windows Task Scheduler), desktop shortcuts, or any other program.

5. **E-mail Alert Messages and Exception Reports** by scheduling exception reports with an option that aborts processing when the report has zero selected records.

6. **Special Capabilities** include:
   - **HTML** exports (including charts & logos) can be embedded in email message bodies.
   - **PDF** exports can include **Color-Coded Bookmarks** & **Table of Contents** for easy online & hardcopy navigation.
   - **PDF** exports can be **password protected** (128-bit or 256-bit AES encryption) and/or **restricted** (for example, prevent editing).
   - **PDF** files can be **Merged** (and bookmarks added) to create mixed page layouts, combine & staple output.
   - **PDF** files can be **Merged to create multiple Layers inside a single pdf file**. Within Acrobat reader, users can then turn the visibility of each layer on or off. This is particularly useful for map layers. You can control the naming and initial visibility of each layer.
   - **PDF** files can be **linearized (web-enabled)** for faster opening from a web url.

- **Export/Burst to BMP, JPEG, WMF, EMF, EPS, PNG, TIF or GIF Image Files** (via an option to convert PDF files to image files).
- **Add web, email, or file links (hotspots) with optional text to PDF files**.
- **Add an image with an optional hotspot to PDF files**.
- **PDF Forms** can be **Filled** (and optionally **flattened**) using Crystal formula values.
- **PDF Exports** can **generate Form Fields** based on formulas placed on the report. This means that **you can use Crystal Reports to design pdf forms, and Visual CUT to generate and distribute these forms**.
- **PDF files with embedded files and drill-down links can be created.**
- Bursting to **Excel** can generate either a separate excel workbook file for each Group or **multiple sheets (tabs) within a single workbook.**
- **Excel** exports can be inserted as tabs in a specified excel file to create ***briefing books.***
- **Excel** exports can display auto filter interface.
- **Excel** exports can be password protected.
- **Excel pivot tables can be generated automatically**.
- **Multi-tab excel exports (due to 65,536 row limitation) can be merged and converted into single-tab .xlsx files**.
- **MS Word** template documents can be filled (mail merged) with Crystal field/formula values and saved/emailed to dynamic destinations.
- **Text** exports can be embedded in email message bodies.
- **Text** exports or files can be **merged** into new or existing files.
- **Export to multiple file formats in a single pass.**
- Print to **multiple printers** in a single pass.
- Print Burst so each Group Level-1 becomes a separate print job.
- **Split the printout** of a pdf file to **different printer trays**.
- **FTP** files to dynamically named web server folders using simple or secure connection.
- **Auto-Refreshing Web Dashboards**
- **Zip and password protect files** for secure single-file emailing of report exports.
- **ODBC** Exports can replace or append to existing tables, This provides ETL (Extract, Transform, Load) and scheduled data snapshot functionality.
- **Capture incoming emails into a database and use them to trigger reports and database updates**.

Install / Remove

Selecting a Version: 8.5, 9, or 11 (XI R2)

In most cases, you should install version XI R2. It can run all .rpt file versions, including 7, 8, 8.5, 9, 10, XI, 2008, 2011, and 2013. Running a Crystal 2008/2011/2013 report in Visual CUT XI R2 is exactly like running it in Crystal XI R2. New features degrade gracefully and the report runs as if it was a Crystal XI R2 report. Note: **You can support new features by delegating exporting/bursting to DataLink Viewer 2011**.

You should install the 8.5 version only if your reports are using an old data source that is not compatible with later versions of Crystal (such as Oracle versions below Oracle 7.3.4).

If you are using Crystal 10, both the 9 as well as XI versions of VC will run your reports (just as Crystal 9 can run Crystal 10 reports). The parameter dialogs in version 9 are the same style as in Crystal 10, and some users don't like the parameter dialog style in XI R2. You can install more than one version on the same machine and compare.

Installation:

Make sure you install while being logged in as an Administrator on that PC.
1. **Remove the old version**: if you have a previous version installed, use the old msi file or the Add/Remove control to remove it.
2. **Extract the msi file to the local hard drive.** Don't run the msi file from within the zip file. Make sure the .msi file is placed on the LOCAL hard drive (not a CD or a mapped drive). Installation (.msi) files have self-repair and on-demand installation (for other users) functionality. Do not delete/move the msi file unless you removed the software.
3. **Install** the software by double-clicking the **.msi** file.

urlmon.dll message

You can ignore and click OK to continue if the installer shows a message about not being able to update urlmon.dll.

Important steps After Installation

In post-XP Windows versions, the application folder (e.g., c:\Program Files (x86)\...) is write-protected by default. When Visual CUT gets loaded for the 1st time, it redirects the key files (Visual CUT.mdb, DataLink_Viewer.ini, ReportList.txt, ReportList.grd) to a MilletSoftware folder under \ProgramData or \AppData. The best way to navigate to this location is to click the Version Information button, and double-click the path information at the bottom of that window. **Be sure to give other users modify permissions to that folder**. Otherwise, Windows will automatically manage a dedicated version of these files for each user under a **virtual store** folder: users\user name\appdata\local**VirtualStore**\Program Files... (this can lead to odd behaviors whereby saved settings are not available to other users or scheduled processing). You can see an explanation of this at: http://www.west-wind.com/WebLog/posts/5584.aspx.

To allow administration of the smtpQ service, you may need to Right-click the Visual CUT.exe and set it (Properties, **Compatibility** tab) to ***Run as Administrator*** (set the option to apply to all users).

PDF Processing Errors

If you get one of these errors: ***Error 429: ActiveX component can't create object , iSED.exe registration error , 'error accessing ole registry' , '*Automation error The system cannot find the file specified' *or 'Out of Memory'***

Some pdf processing options require the use iSED.exe as a component, which must be registered while you are logged in as an Administrator.

Use Notepad to place the following command line in a batch (.bat) or command (.cmd) file:

"c:\Program Files\Visual CUT 11\ised.exe" -regserver

or, on a 64-bit machine:

"c:\Program Files (x86)\Visual CUT 11\ised.exe" -regserver

Right-click the batch file and select **'Run as administrator'**

Avoid Installation on a Machine acting as a Crystal Reports Server (Crystal Reports Server, BOE, SBS, MAS90, SAGE 100):

Due to the risk of runtime component conflicts, you should avoid installing the software on a machine acting as a Crystal Reports Server.

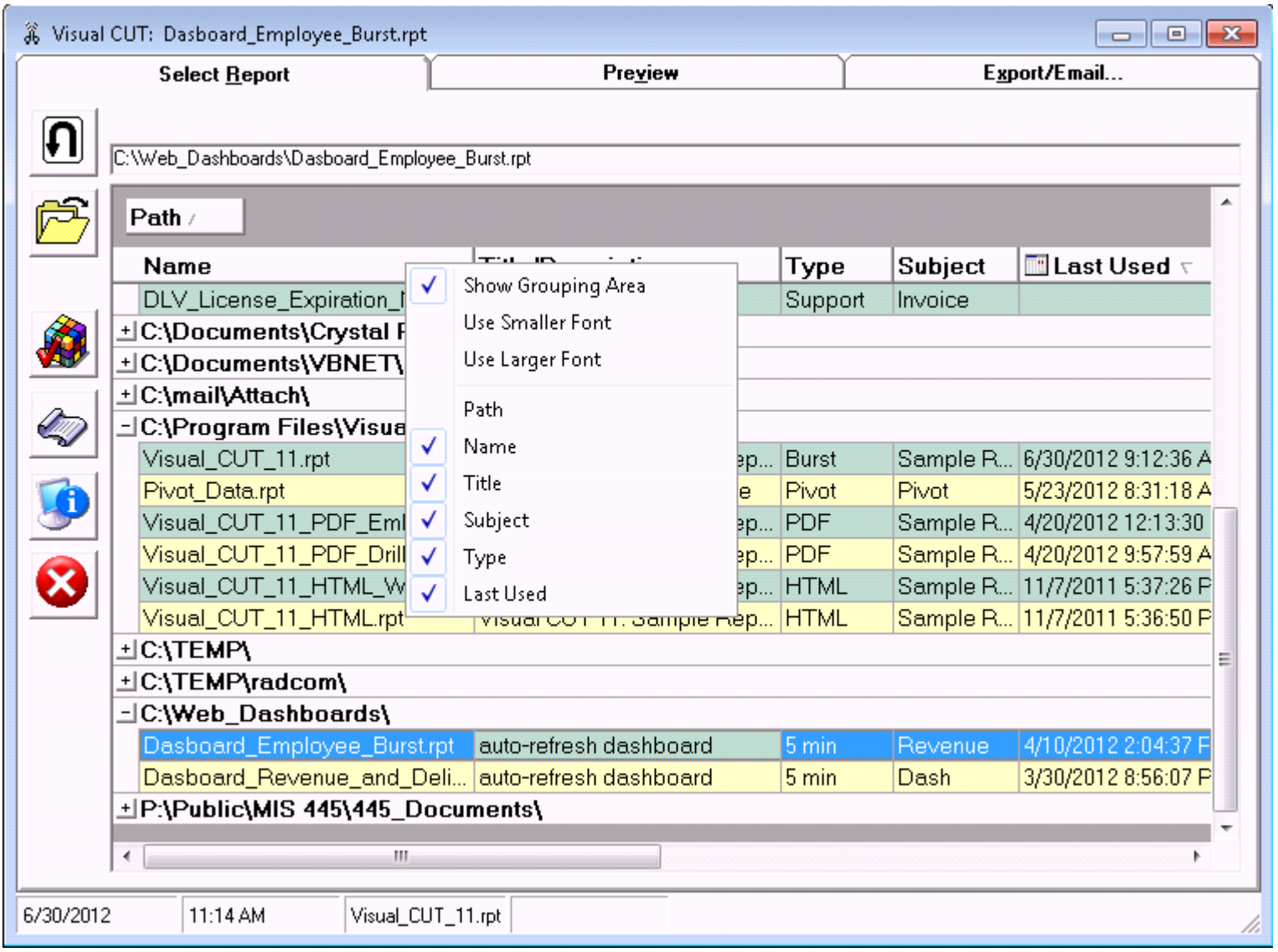
Changing a Version (between 8.5, 9, or XI):

If you installed one version and then you decide to REMOVE it and install another version, be sure to first rename the old **Visual CUT.mdb** file, since it is not identical across all versions. To transfer settings from an old Visual CUT.mdb to the new one, follow these steps:

1. Open both Visual CUT.mdb files (old and new) in MS Access and **copy all tables except for Export_Opt from the old to the new**.
2. If you installed to a new folder, copy **ReportList.txt**, **ReportList.grd** and **DataLink_Viewer.ini** from the old folder to the new one.

Step 1: Select Report

After starting Visual CUT, you would see the following screen:



Use the button to browse for and open a report for the first time. Previously selected reports are listed in a grid and can be launched by **double-clicking** or by **Right-Clicking** and selecting 'Preview Report' from the **popup menu**.

Use the button to reload a report (if changed and saved in Crystal).

To **resize the grid columns**, drag the borders between the column headers.

To **delete a report from the grid** (but not from the hard drive) right-click a row and select 'Delete Row' from the popup menu. The grid information in maintained in a plain text file (**ReportList.txt**).

Use the button to access a dialog for setting various **options**. These options are maintained in the file **DataLink_Viewer.ini**

Use the button to open this **User Manual** as a pdf file.

Use the button to access a dialog with **Version (and system) information**.
That dialog also has a button that allows you to check for software patches on my web site and install them online. This patching mechanism is very fast since the patches are typically very small (contain only file changes).

Use the button to **exit** Visual CUT.

To **delete a report from the grid** (but not from the hard drive) right-click a row and select 'Delete Row' from the popup menu. The grid information in maintained in a plain text file (**ReportList.txt**).

By dragging, clicking, or right-clicking the grid column headers you can apply various options such as grouping the report by any column(s), hiding/showing any columns, sorting, increasing/decreasing font size, etc.

When adding a report to the grid, Visual CUT populates the **title** and **subject** columns automatically if it finds that information in the **summary information** for the report.
You set that information for the .rpt file in Crystal (under the file menu).

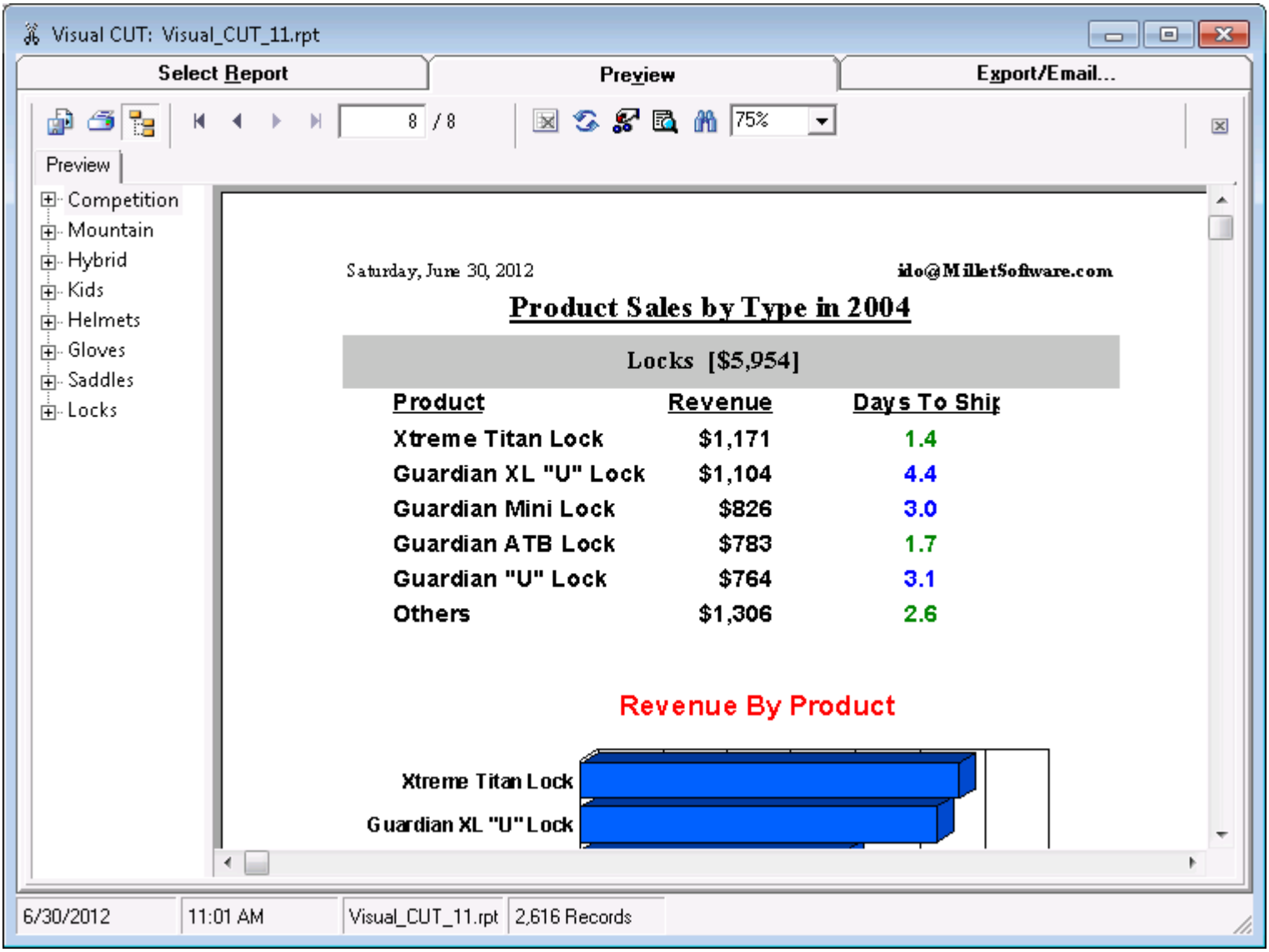
Let's launch one of the sample reports (installed under the **c:\Program Files\Visual CUT** directory in typical installations). You can do this by

1) **Selecting that row and clicking the Preview Tab**, or
2) **Right-Clicking that row and selecting Preview**, or
3) **Double-clicking the row**.

Note: the sample reports expect to find the Xtrete Sample Database (and an ODBC DSN for connecting to that database) on your PC. The sample reports are named so that you know which version of the sample database is expected (8.5, 9, etc.)

Step 2: Preview

The Preview Tab is an integrated viewer that would prompt you for any report parameters and logon information required by the report.



Speeding Up Report Previews

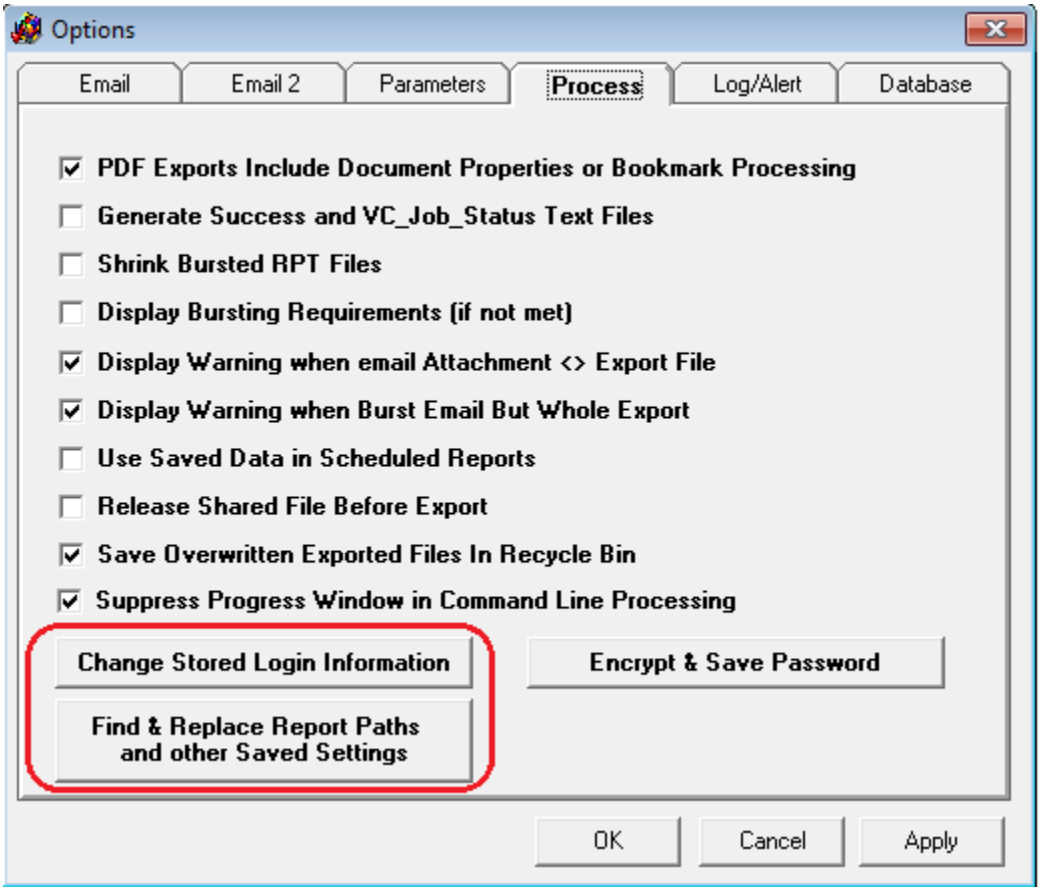
If you are opening a report in Visual CUT, you may wish to cut the time it takes to preview the report so you can get to the business of changing some settings. This can be achieved by saving the report with data in Crystal. When you open a report with saved data in Visual CUT, you get the option to preview the report without refreshing the data.

Changing Processing Options without Previewing

All processing options are stored in Visual CUT.mdb. You can use MS Access to open that database and edit processing options in the **Report_Opt** and **Report_Export_Options** tables.

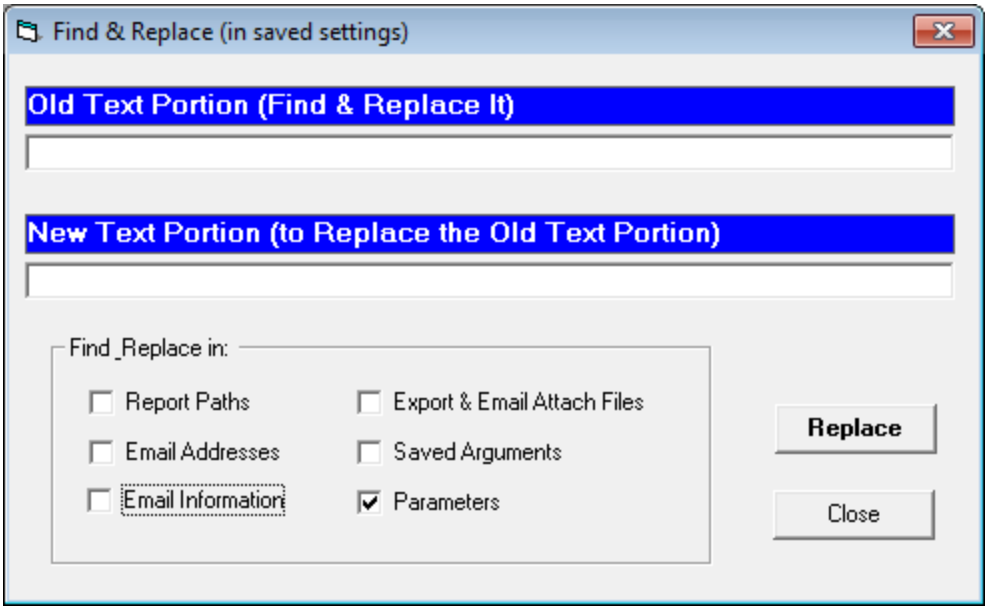
Changing Login, Report Paths & Other Settings without Previewing

Visual CUT stores login information (strongly encrypted) for each report in the Login_Opt table within Visual CUT database. Imagine your password to a data source used by 20 reports has changed. Instead of previewing each of these reports just so that the new logging information is captured, use the **Options** dialog, **Process** tab, to '**Change Stored Login Information**' for all reports. Similarly, use the '**Find & Replace Report Paths and other Saved Settings**' button to update Report Paths (in the database as well as in the grid) or other saved settings.


Options
Email
Email 2
Parameters
Process
Log/Alert
Database
PDF Exports Include Document Properties or Bookmark Processing
Generate Success and VC_Job_Status Text Files
Shrink Bursted RPT Files
Display Bursting Requirements (if not met)
Display Warning when email Attachment <> Export File
Display Warning when Burst Email But Whole Export
Use Saved Data in Scheduled Reports
Release Shared File Before Export
Save Overwritten Exported Files In Recycle Bin
Suppress Progress Window in Command Line Processing
Change Stored Login Information
Encrypt & Save Password
Find & Replace Report Paths and other Saved Settings
OK
Cancel
Apply

Find & Replace Report Paths and other Saved Settings

This dialog allows you to search specific types of saved setting for string replacements.
For example, as shown below, if you turn on the 'Parameters' checkbox, only saved parameter values would be targeted for Search & Replace:



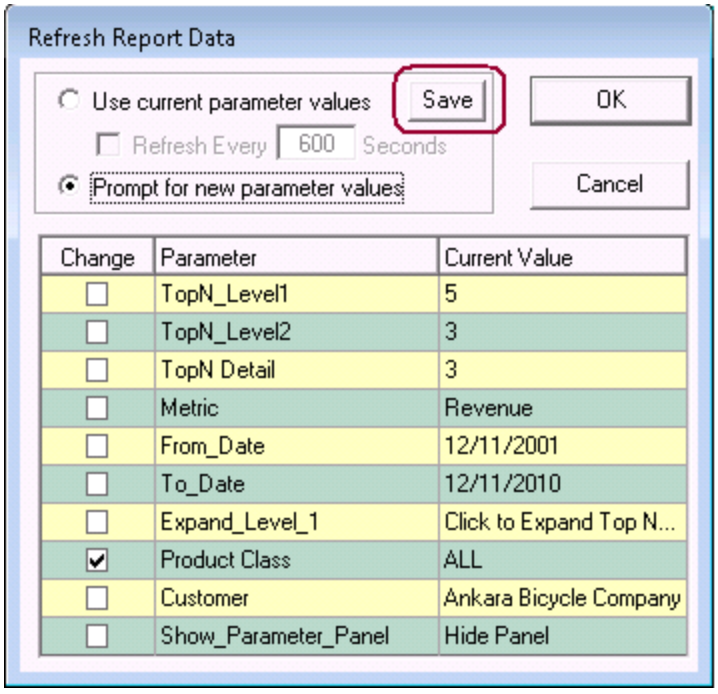
Disabling Find & Replace Categories

In some scenarios you may wish to block the user from some or all of the Find & Replace categories. To do so, you can set the following entry in the **[Options]** section of a **Master_DataLink_Viewer.ini** file **in the application folder**:

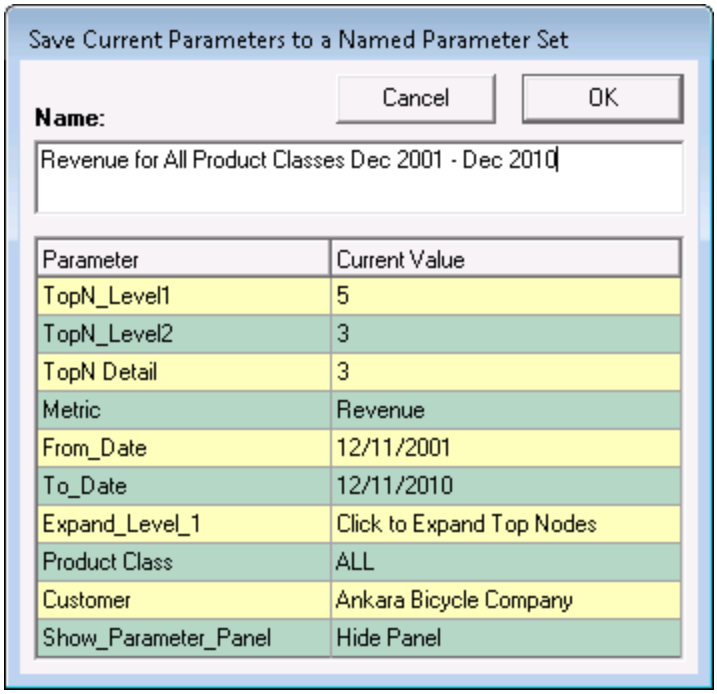
[Options]
// Use **ALL** some combination of: **Report_Paths**, **Email_Addresses**, **Email_Information**,
// **Export_and_Email_Attach_Files**, **Saved_Arguments**, and **Parameters** delimited by '||'.
// For example, the following entry would disable only 2 categories:
Disable_Find_and_Replace_Categories= Email_Addresses||Email_Information

Save and Reuse Named Parameter Sets

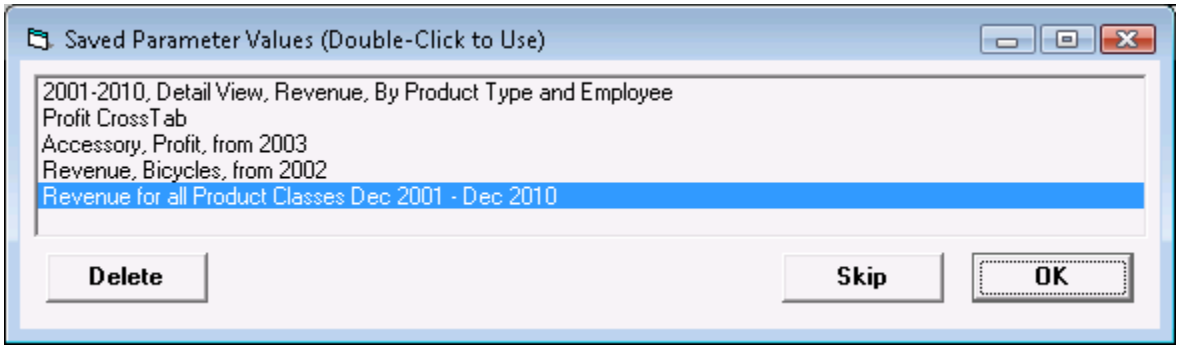
If you click the refresh button for a report that has at least 8 parameters (Options dialog allows you to change that number), a ***Save*** button becomes visible in the following dialog:



If you click ***Save***, the following dialog allows you to save the current parameters value set to DataLink_Viewer.ini under a unique name for this report.



The next time you load this report, you would get a dialog that allows you select and reuse any of the saved named parameter sets for this report:


Saved Parameter Values (Double-Click to Use)
2001-2010, Detail View, Revenue, By Product Type and Employee
Profit CrossTab
Accessory, Profit, from 2003
Revenue, Bicycles, from 2002
Revenue for all Product Classes Dec 2001 - Dec 2010
Delete
Skip
OK

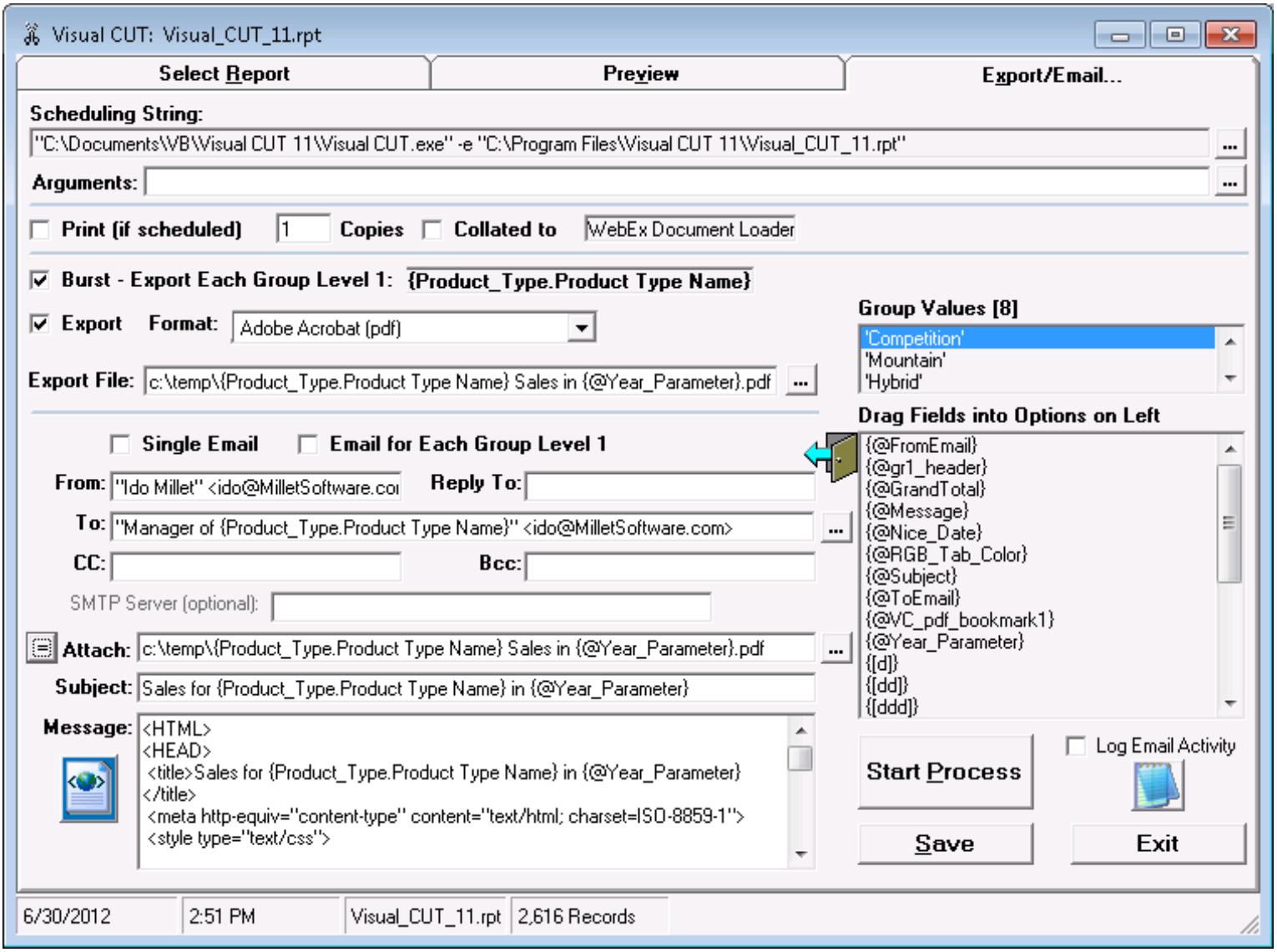
Double-clicking any of the entries (or selecting an entry and clicking the OK button) would launch a dialog allowing you to reuse that set of saved parameter values or selectively change some of these saved parameter values.

This functionality was designed to address scenarios where reports with many parameters are used in one of several parameter patterns. By saving and naming these patterns, the user can reuse them.

Note: if you are a report developer, you can deliver these saved parameter patterns to a user machine by copying the [Named_Parameter_Sets] section in your DataLink_Viewer.ini file to the user's DataLink_Viewer.ini file.

Step 3: Export/Burst/Email…

The 3rd tab of the application is where the real magic occurs. Let's review how each area on this screen is used.

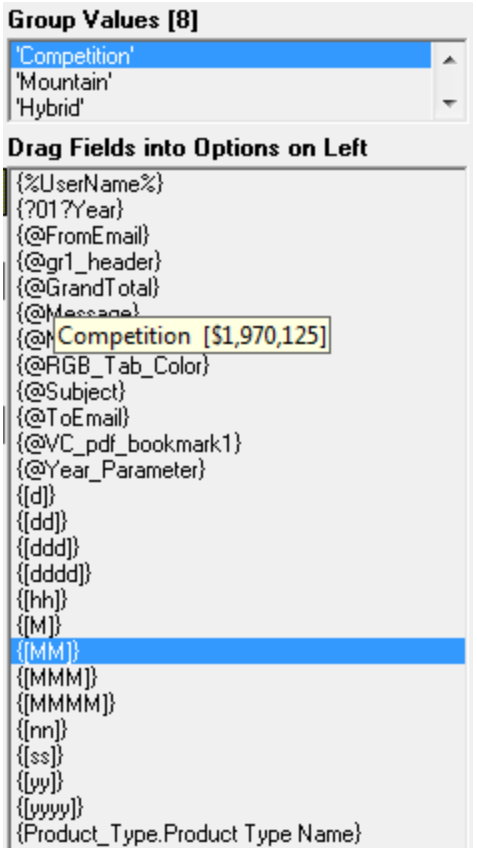


Group Values Area

Visual CUT automatically detects Group Level 1 in the selected report and, if grouped on a Text or Numeric field/formula, populates this area with all group values. The name of the Group Level-1 field or formula is presented at the top of exporting options area, where the exporting burst option can be activated.

Note: the Group Values area is visible only if Visual CUT can burst the report. This boils down to two requirements: 1. The report must be grouped at level 1 on a Text or Numeric field/formula and 2) that field/formula must be placed (it can be suppressed) on a Group Level 1 Header/Footer (the section can be suppressed).

Fields & Formulas Area



Visual CUT automatically detects and lists (sorted alphabetically) in this area all fields and formulas that are placed (even if suppressed) in **Report** or **Group Level-1** Headers and Footers (even if the sections are suppressed). Hovering your mouse over a field provides a tooltip reflecting the dynamic value associated with that field for either
a) the whole report (if in a report-level section), or
b) the selected group in the Group Values area (if in a Group1 section)
For example, hovering over the {@gr1_header} formula produced the tooltip shown in the image provided here.

You can drag-and-drop these fields & formulas into the processing options area to specify dynamic options (options that change their values based on the report or group being processed).

Also listed in that area are main report parameters that are in use. For example, **{?01?Year}** is the {?Year} parameter. The 01 indicates the position of the parameter in the list of parameters. That position is important in constructing parameter arguments such as "Parm**1**:2015". The dynamic string values provided for parameter fields are converted from the actual data type of the parameter (Boolean, date, datetime, time, number, currency, or string). Multi-value parameters are returned as comma separated values. Range parameters are returned as [from-to] strings.
If you need more control over the values returned from parameters, create a formula that returns the parameter value in the desired format (typically using the ToText() function), and place the formula (suppressed) in the report header or footer.

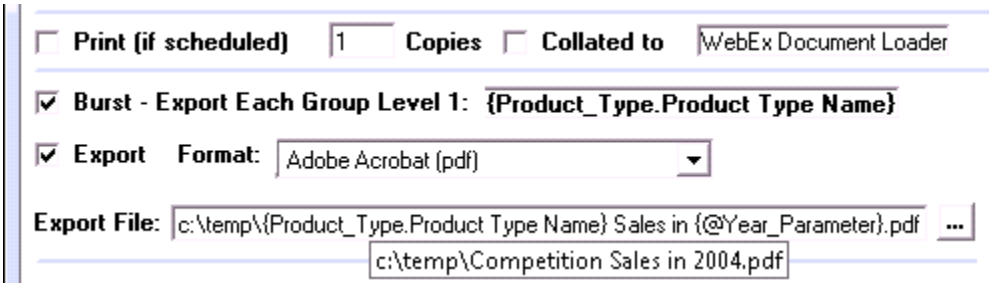
Since adding current day, month, and year values to options such as export file names and email subject are common needs, Visual CUT includes in the Fields & Formulas area a list of **current date values** (day, month, year) with various variations (number, name, length). This allows you to avoid creating formulas in your report just to pass current date values to Visual CUT.

Other fields listed in this area include:
{%UserName%} – provides the Windows User ID
{[ODBC_DSN:]} – provides the ODBC DSN used to override the default DSN for the report.

Notes:

- Text objects containing fields/formulas are not included in this process.
- Even if the field/formula is suppressed it will be recognized.
- Even if the report section is suppressed, the fields/formulas will be recognized.
- **Report** Header/Footer subsections a,b,c,d,and e participate in field/formula detection.
- If a **Group** Header/Footer Level-1 has subsections, only the **first and second** (GH1a, GH1b, GF1a, GF1b) subsections participate.

Exporting/Bursting/Printing Options Area



This area allows you to specify the Exporting format (all file export formats are available) and whether you wish to export the whole report or burst each Group Level-1 into its own file.

When bursting, you should typically specify a dynamic ***Export File Name*** option by dragging fields/formulas from the report into the export file name. This would ensure that each group export results in a unique and meaningful file name. As demonstrated in the image above, **if you hover your cursor over any option, a tooltip displays the dynamic value corresponding to the currently selected group** (in the Group Values area). So,
`c:\temp\{Product_Type.Product Type Name} Sales in {@Year_Parameter}.pdf`
gets a tooltip of:
`c:\temp\Competition Sales in 2004.pdf`

Note: when exporting, **if the target file exists, it is first deleted and moved to the Recycle Bin.**

Exporting to Multiple Files/Formats In a Single Pass

Imagine that you need to export and email a report in two or more file formats. Visual CUT allows you to do this in a single pass. You can specify multiple export file names (separated by a semi-colon) and Visual CUT will export to all of them without retrieving the report multiple times. For example: c:\temp\test.xlsx;c:\temp\test.pdf

The multiple export file names can either be entered into the export file name option or specified in the "Export_File" command line argument.

Visual CUT automatically uses the appropriate export format for files with different extensions (**.pdf**, **.xls**, **.doc**, **.htm**, **.txt**, **.csv**, or **.rpt**) than the extension of the first file name.

The first export file extension should match the export format selected for the report. Since some export formats, such as **.xls** and **.txt**, have extra options, it makes sense to select that export format as the first file to be exported (allowing saved settings to control the extra options). Export formats that have no variations (such as **.pdf** and **.rpt**) should be specified as the extra export files.

Note: this applies even to bursting, so each bursting cycle can generate multiple exports.

Replacing Illegal Characters in Dynamic File Names

Starting November 2011, **you no longer need to worry about illegal and non-printing characters in file names** unless:

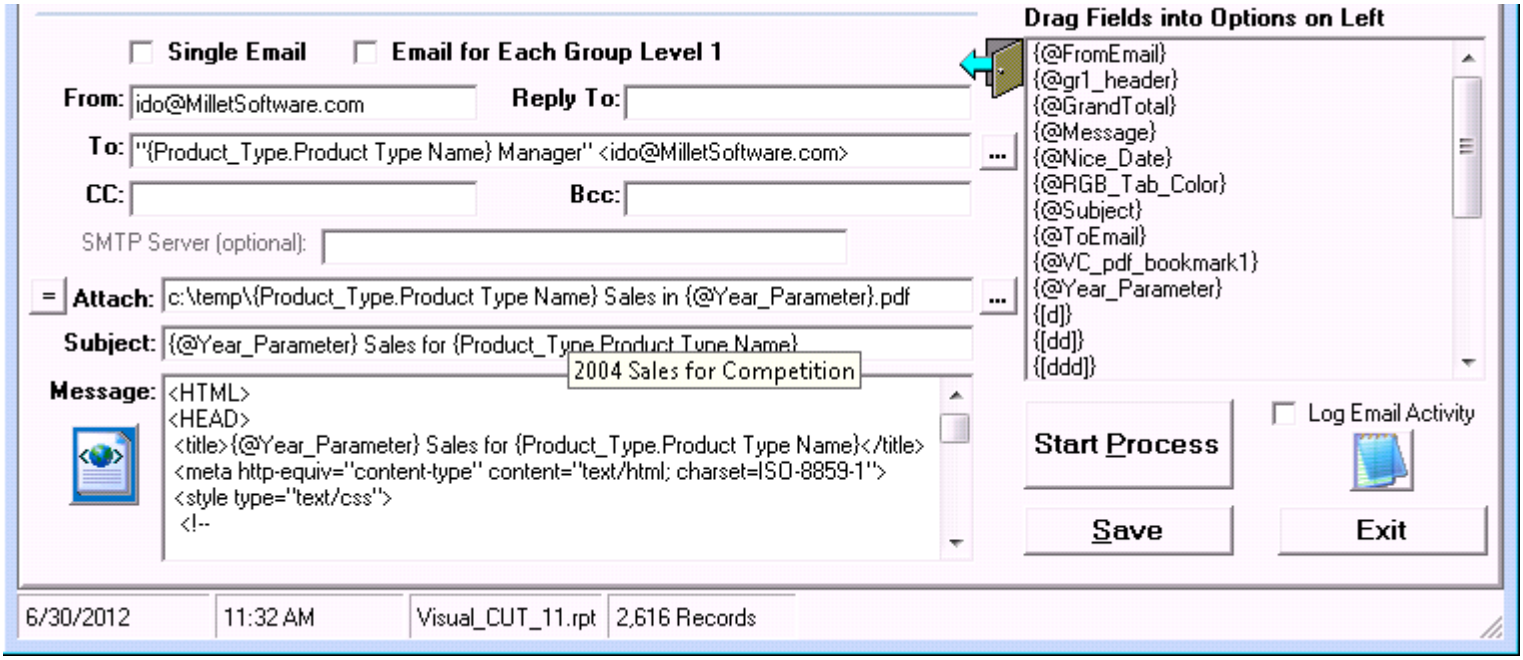
a) your export file names contain the illegal characters '*****' or '**?**' (those characters are not replaced because they may be used in wild card selection of email attachments)
b) you wish to apply your own logic for character substitution

Visual CUT takes care of replacing illegal characters in export file names, email file attachments, and .eml files (.eml files are used in email queuing). Visual CUT also removes non-printing characters from these options. The logic of illegal character substitutions is shown in the sample formula below. When hovering your mouse over the export file name or email attachment option, the tooltip shows the corrected dynamic value.

If you need to override the automated handling and control what legal characters substitute for what illegal characters, you can use a Crystal formula like the one shown below. Remember to place that formula in the RH/RF or GH1/GF1 section of the report so that it becomes available for drag & drop within Visual CUT.

```
// illegal Characters for windows file names include: ?/\|<>:"*
// Note: change InputFileName to your file name field or formula
Local StringVar InputFileName := "Test?/\|<>:""*" ;
Local StringVar OutputFileName ;
OutputFileName := Replace(InputFileName, "?", "_");
OutputFileName := Replace(OutputFileName, "/", "_");
OutputFileName := Replace(OutputFileName, "\", "_");
OutputFileName := Replace(OutputFileName, "|", "_");
OutputFileName := Replace(OutputFileName, "<", "{");
OutputFileName := Replace(OutputFileName, ">", "}");
OutputFileName := Replace(OutputFileName, ":", "_");
OutputFileName := Replace(OutputFileName, """", "'");
OutputFileName := Replace(OutputFileName, "*", "x");
// If destination is web server, replace spaces with "_"
//OutputFileName := Replace(OutputFileName, " ", "_");
```

Email Options Area



If you wish to generate a single Email from the selected report, turn on the "***Single Email***" option. If you wish to generate an Email message for each Group Level-1, either in conjunction with electronic bursting (Area 4) or as a simple mass e-mailing process, turn on the "***Email for Each Group Level 1***" option. Of course, you may elect to turn off both options.

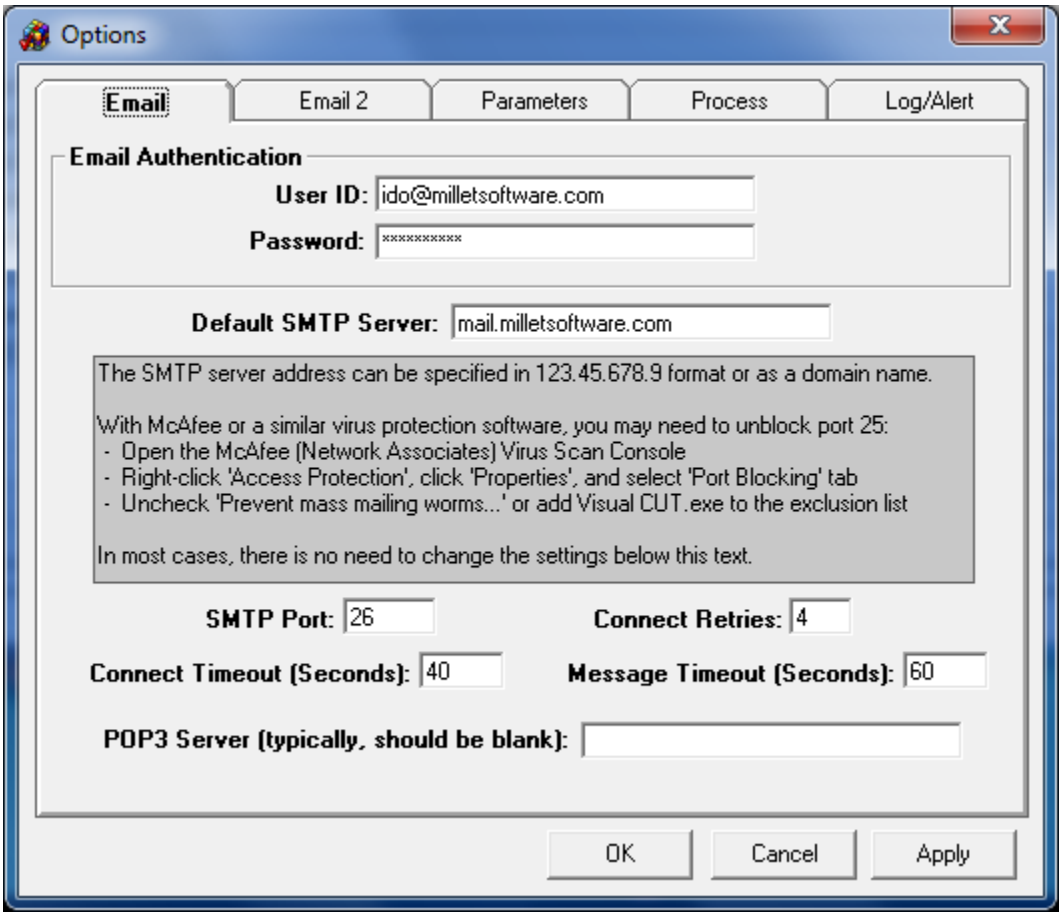
Combining Static & Dynamic Content

In all options, you can combine static text and dynamic values you drag from the fields/formulas area on the right. Moving the cursor over any of these options results in a tooltip display combining the static text with the dynamic content of the fields/formulas (based on the selected Group Value).

For example, in the example shown above, the email subject line resolved the text:
{@Year_Parameter} Sales for {Product_Type.Product Type Name}
into the dynamic content of:
2004 Sales for Competition
This allows you to see what dynamic content you are building into your processing options.

Specifying SMTP Server (optional)

In most cases, you should leave the *SMTP Server* option blank, Visual CUT will use the Default SMTP Server address that was set in the global Options dialog:



Embedding Report as Image in Email Message Body

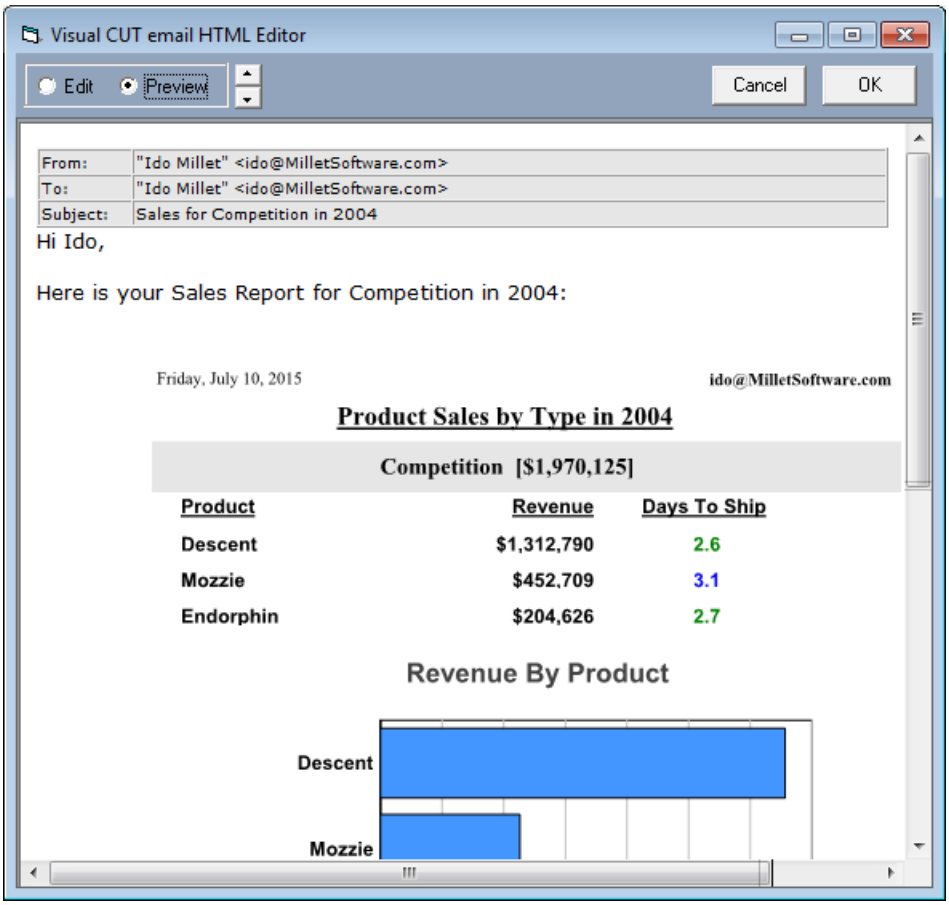
Imagine you need to embed an image of the report's 1st page inside the email message body. The main advantage of that approach is that the recipient would see the report in the email message without needing to open an attachment. Here are the steps to achieve this:

1. Export the report to pdf and use **PDF_Save_As** to convert to an image. For example, bursting to: **c:\temp\Sales for {Product_Type.Product Type Name} in {@Year_Parameter}.pdf** the argument would be:
   "**PDF_Save_As**:c:\temp\Sales for {Product_Type.Product Type Name} in {@Year_Parameter}.pdf> c:\temp\Sales for {Product_Type.Product Type Name} in {@Year_Parameter}.png>**PNG**>72"
   Each page is saved with page number added at the end, so the 1st page from Sales for Competition in 2004.pdf would create a png file called: Sales for Competition in 2004**1**.png

2. Embed a reference to the png image file inside the HTML message body. For example:

```
…<BODY>
Hi Ido, <br><br>
Here is your Sales Report for {Product_Type.Product Type Name} in {@Year_Parameter}: <br>
<IMG src="file:///C:/TEMP/Sales for {Product_Type.Product Type Name} in {@Year_Parameter}1.png">
</BODY></HTML>
```

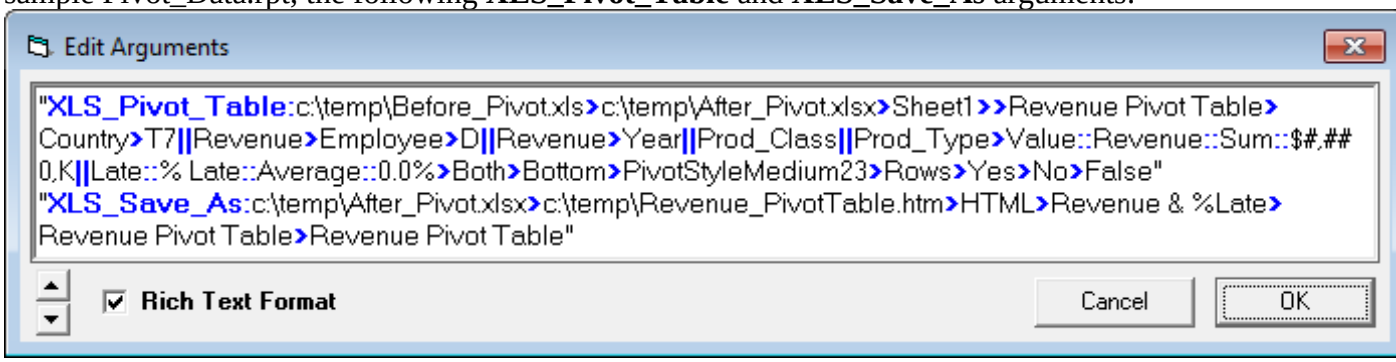
→



Embedding Pivot Tables in Email Message Body

To embed a Pivot Table inside the email message body, follow these steps:

1. Use **XLS_Save_As** to target the pivot table for conversion to HTML. For example, using the sample Pivot_Data.rpt, the following **XLS_Pivot_Table** and **XLS_Save_As** arguments:

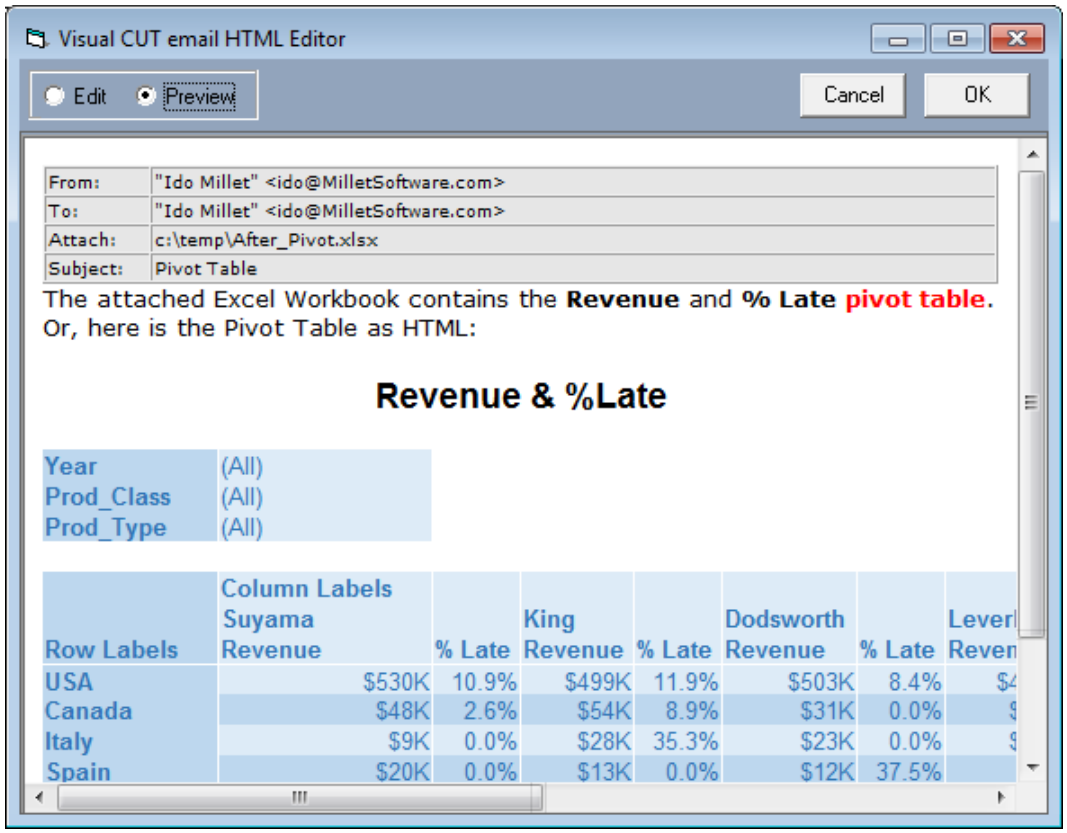


   would generate the **After_Pivot.xlsx** workbook with a tab called **Revenue Pivot Table** and a pivot table with the same name. The **XLS_Save_As** argument would then save the pivot table into an html file (**c:\temp\Revenue_PivotTable.htm**).

2. Embed a reference to the html file inside the HTML message body. For example:

```
<BODY>The attached Excel Workbook contains the <STRONG>Revenue</STRONG> and <STRONG>% Late</STRONG> <STRONG><FONT color=#ff0000>pivot table</FONT></STRONG>.
Or, here is the Pivot Table as HTML: [[Insert_File:c:\temp\Revenue_PivotTable.htm]] </BODY></HTML>
```

→



Embedding HTML Export in Email Message Body

Imagine you need to export a report to HTML but you wish to embed the resulting content (including charts and images) inside the email message body. To do this you need to observe the following simple rules:

3. The email message body must be left completely blank.
4. The first attached file name must be the html file to embed as the HTML email message (typically, it's the report export to HTML). That file name must end with **.htm** or **.html**
5. For the Crystal 9 or 11 version of Visual CUT, use **HTML 40** as the export format
6. For the Crystal 8.5 version of Visual CUT, use **HTML 32** as the export format

Limitations:

- email clients typically corrupt the formatting of complex HTML messages when users attempt to Reply to or Forward the email message.
- **Outlook 2007** is very limited in rendering HTML content. Use an older version of Outlook if you need HTML embedding in email messages or instruct Outlook 2007 users to open HTML emails in a full window (double click), and then click the "Other actions" toolbar button and choose "**View in browser.**"

To avoid these limitations, a different method of sending report content as HTML is described in the following sections:

Sending Message Text as HTML

Embedding Images inside the HTML email body

Dynamic Tables inside HTML Email Messages

Using Cascading Style Sheets (CSS) in HTML messages

Embedding TEXT Export in Email Message Body

Imagine you need to export a report to Text but you wish to embed the resulting content inside the email message body. To do this you need to observe the following simple rules:

1. The email message body must be left completely blank.
2. The text file (probably the file exported by Visual CUT, but it can be any other file as well) to embed in the message body must be either the **only** specified attachment or the **first** file in the attachment list (reminder: multiple file attachments are separated by a ";" without spaces).
3. The text file name must end with **.txt, .prn, or .text**

Embedding File(s) Content in Email Message Body

Imagine you need to insert a standard header and footer into the email message body, or you wish to give the user an easy way to modify the content of a specific paragraph by editing a text file, or you are specifying the email body via a command line and wish to refer to the body content via a file rather than by including the full content in the command line itself. To do any of the above, you can embed within the message a references to a file using the following structure: **<<Insert_File:File_Path_and_Name>>**
For example, **<<Insert_File:c:\temp\MyFile.htm>>**

1. For HTML email messages, you should use **[[Insert_File:** … **]]**
2. When Visual CUT encounters such a "file token" it replaces it with the content of the specified file (if such a file exists).
3. **References to report fields/formulas** are replaced with their **dynamic content**.
4. You can use **as many file tokens as you wish** within a single message body.
5. You can **use a Crystal formula** that results in such a token and embed the reference to that formula in the message body. Visual CUT would first resolve the formula reference into the token and then replace the token with the file content. A typical use for this would be to embed a different paragraph in an email message depending on the number of days an account is overdue.
6. If an inserted file has embedded file tokens, they would be replaced as well (the process is **recursive**).

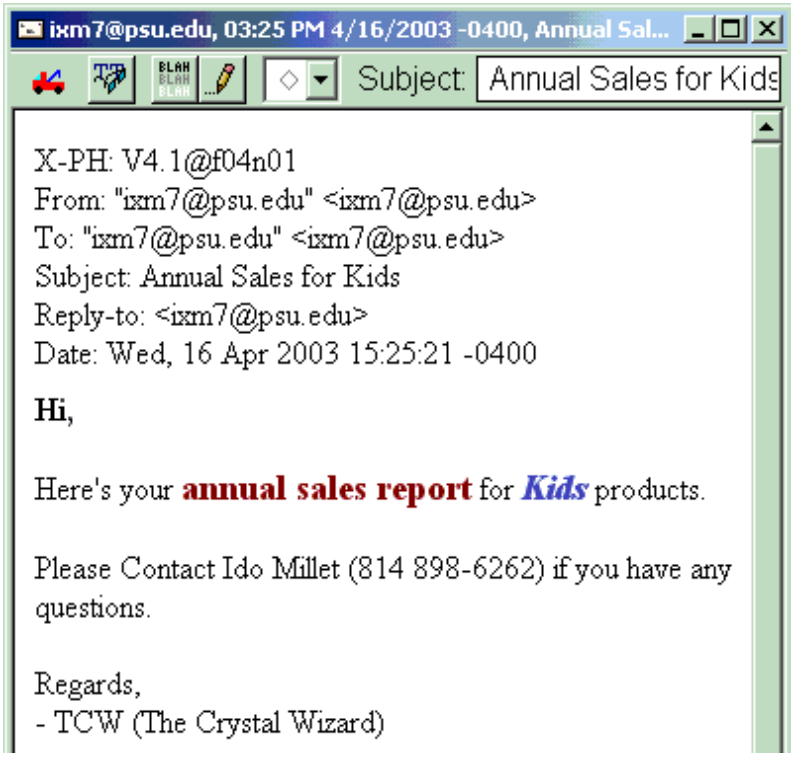
Sending Message Text as HTML

Visual CUT ensures that e-mail message text that starts with <HTML> and ends with </HTML> is interpreted as HTML when received as e-mail by the recipients. This allows you to use HTML tags to create a message that displays with a variety of nice formatting effects.

You can use the integrated HTML editor to create the message and drag and drop dynamic content into appropriate locations within the resulting HTML text.

```
<html>
<b>Hi, </b> <br>
<br>
Here's your <big><font color="#990000"><b>annual sales report</b></font></big>
for <big><b><i><font color="#3333ff">{Product_Type.Product Type
Name}</font></i></b></big>
products.<br>
<br>
Please Contact Ido Millet (814 898-6262) if you have any questions.<br>
<br>
Regards,<br>
- TCW (The Crystal Wizard)
</html>
```

The message body above resulted in the following e-mail message body:



Integrated HTML Editor

Just to the left of the email message body, you can click on button to start an integrated HTML editor. If the message is not already in HTML syntax, clicking the button will add the necessary HTML syntax including a few useful cascading style sheet options (defaulting the message body text as well as any Table text to Verdana).

The editor allows you to easily change formatting, insert tables, images and links, check spelling, and edit the text, all from an intuitive word processing interface. It also allows you to preview the resulting email messages with dynamic values substituted for Crystal field/formula references.

You can copy and paste content into the HTML editor while preserving formatting from other software such as Microsoft Word or Excel. If you copy content from excel, it becomes an HTML table, so instead of starting a table from scratch, you can create it in Excel, and then copy into the Visual CUT integrated HTML editor.

Embedding Images inside the HTML email body (old email engine – no longer available)

In order to include an image as an embedded file, the HTML syntax of your message would include something like:
** <img border="0"**
src="file:///C:/Program%20Files/Visual%20CUT/logo.gif"
**width="211" height="69"> **

You would need to attach that file to the email message.
The attachment would be specified as something like
C:\Program Files\Visual CUT\logo.gif
(separated from other attachment by ";")

You don't need to attach the image if it is a reference to an image available on the web.

Embedding Images inside the HTML email body (2009 email engine)

If you are using the GUI HTML editor for the email message body, just insert an image and Visual CUT would take care of embedding it in the email message.

If you are manually creating the reference to the image, use the following structure:
<IMG src="file:///image_file_path_and_name**">**
For example, **<IMG src="file:///**C:/Pictures/My_Logo.jpg**">**

The path and name of the image file must point to an existing file. Of course, it may include a dynamic reference to a Crystal field or formula. For example:
<IMG src="file:///C:/Pictures/{Product_Type.Picture_Path}.jpg**">**

You should not specify the image file as an attachment.

Dynamic Tables inside HTML Email Messages

See Video demonstration.

HTML tables are very useful, particularly when sending email messages to mobile devices. Because the width of an HTML table and its columns can be specified as percentages of the available screen width, the information would adapt itself to the device displaying the message.

It is very easy to create static HTML tables (use the integrated HTML editor). However, **this section describes a powerful technique for using Crystal formulas to create dynamic HTML tables (with information from your Crystal report). You can then embed the formula containing the full table syntax in the email message body**.

The general idea is to use:
1) one formula (in a Report or Group header section) to set the value of a string variable to the **table header** syntax (**<Table>**….
2) ↻ another formula (in a Detail or Group section) would repeatedly append table rows
3) a final formula (in a Report or Group Footer section) would provide the table footer and close the table syntax (using …**</Table>**)

The Visual CUT 11.rpt sample report demonstrates this technique (and email bursting) using: **HTML_Table_1_GH1 (formula placed in GH1)**:

```
// In order to capture multi-section detail into an HTML table string
// (called HTML_Table), this report uses 3 Formulas:
// HTML_Table_1_GH1 sets the variable to the Table Header info
// HTML_Table_2_GF2 appends a row of data (for each Group Footer level 2)
// HTML_Table_3_GF1 Closes the Table

// Place this formula in GH1 to set formatting, column names, widths, etc.
WhilePrintingRecords;
Stringvar HTML_Table;
// Reset the string variable only if this is not a repeated group header
IF NOT InRepeatedGroupHeader THEN
HTML_Table := "<table width=""100%"" border=""1"" cellpadding=""5"" cellspacing=""0""
bordercolor=""#0033CC"">" +
        "<tr bgcolor=""#66CCFF"">" +
          "<td width=""60%"" align=""left""><strong><big>Product</big></strong></td>" +
          "<td width=""20%"" align=""right""><strong><big>Revenue</big></strong></td>" +
          "<td width=""20%"" align=""center""><strong><big>Days<br>To Ship</big></strong></td>" +
        "</tr>"
```

HTML_Table_2_GF2 (placed in GF2):

```
// Place this formula in GF2 to append Table Row Information
WhilePrintingRecords;
Stringvar HTML_Table;
HTML_Table := HTML_Table +
"<tr>" +
// Here we add the Product Name column
// (note that where we need to have " we must specify "")
"<td><div align=""left"">" +
{Product.Product Name} +
"</div></td>" +
// Here we add the total Revenue column:
"<td><div align=""right"">" +
"<b>$" + ToText(Sum ({@value}, {Product.Product Name}),0) + "</b>" +
"</div></td>" +
// Here we add the Average Days to Ship:
"<td><div align=""center"">" +
(
IF Average ({@Days_To_Ship}, {Product.Product Name}) > 5 THEN
// Slow shipping, so format as Red
"<b style=""color: rgb(255, 0, 0);"">" +
ToText(Average ({@Days_To_Ship}, {Product.Product Name}),1) +
"</b>"
ELSE IF Average ({@Days_To_Ship}, {Product.Product Name}) < 3 THEN
// Fast shipping, so format as Green
"<b style=""color: rgb(4, 180, 4);"">" +
ToText(Average ({@Days_To_Ship}, {Product.Product Name}),1) +
"</b>"
ELSE
// Intermediate level of performance. Format as Blue
"<b style=""color: rgb(0, 0, 255);"">" +
ToText(Average ({@Days_To_Ship}, {Product.Product Name}),1) +
"</b>"
)  +
"</div></td>" +
"</tr>"
```

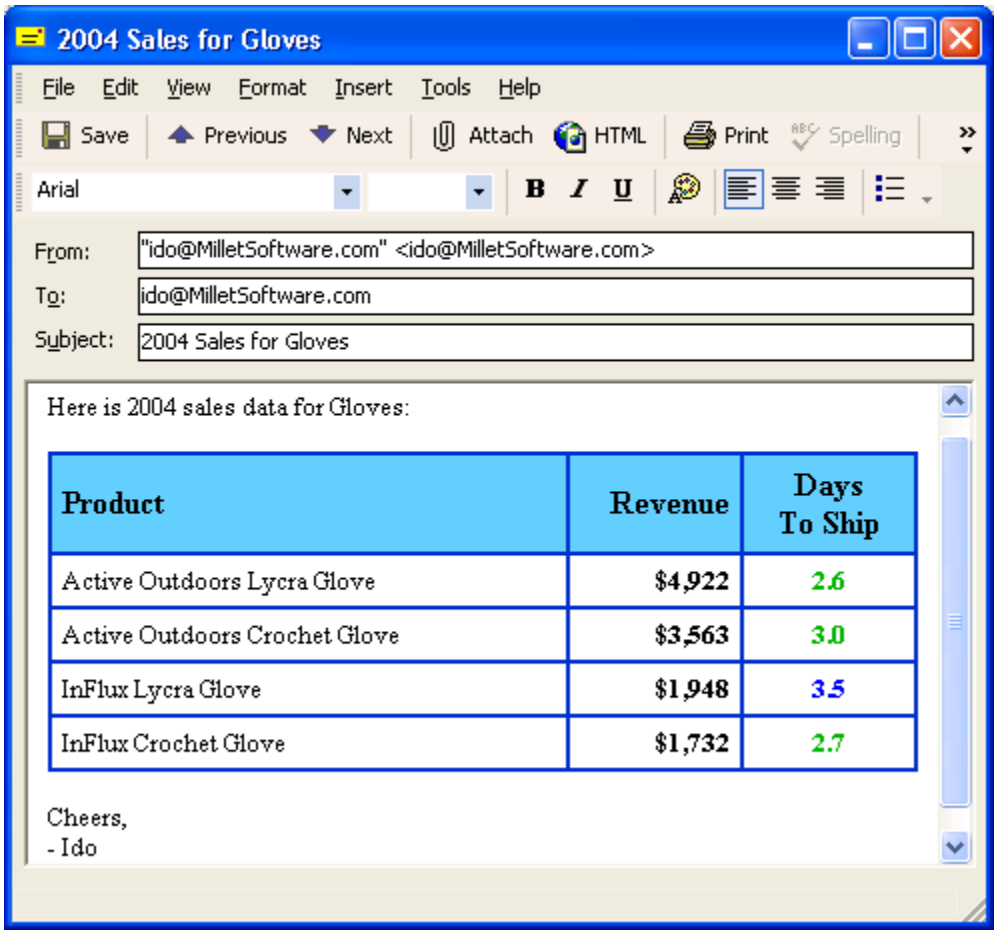
HTML_Table_3_GF1 (used in Visual CUT to embed the HTML Table in email body):

```
// Place this formula in GF1 to "close" the HTML Table
WhilePrintingRecords;
Stringvar HTML_Table;
HTML_Table := HTML_Table + "</table>"
```

In Visual CUT, the report is set to burst and the **{@HTML_Table_3_GF1}** formula (because it was placed in a GF1 section) is available for drag & drop into an email message body:

```
<HTML>
Here is {@Year_Parameter} sales data for {Product_Type.Product Type Name}:<br><br>

{@HTML_Table_3_GF1}<br><br>
Cheers,<br>
- Ido
</HTML>
```

The resulting email message provides the dynamic content with an embedded HTML table:



Applying Alternating Row Color

To create an alternating background color effect to the HTML table rows, use the **bgcolor** attribute of the **tr** (table row) html tag.
For example, the following HTML email message used the same approach described in the previous section, except that instead of a static "<tr>" to start each data row (GH2 formula), a string variable called **ls_alternating_color_TR** is used to alternate between "**<tr>**" (no color) and "<tr bgcolor=""#E0FFFF""> " (a light blue background color):

```
WhilePrintingRecords;
Stringvar HTML_Table;
// instaed of using "<tr>" to start each row, alternate row colors between "<tr>" and "<tr
bgcolor=""#E0FFFF"">"
Stringvar ls_alternating_color_TR;
IF ls_alternating_color_TR = "<tr>" THEN
      ls_alternating_color_TR := "<tr bgcolor=""#E0FFFF"">"
ELSE
   ls_alternating_color_TR := "<tr>" ;

HTML_Table := HTML_Table +
Chr(10) + ls_alternating_color_TR + …
```

From:	"Ido Millet" <ido@MilletSoftware.com>
To:	"Locks Manager" <ido@MilletSoftware.com>
Attach:	c:\temp\Locks Sales in 2004.htm
Subject:	Locks Sales in 2004

Here is **2004** sales data for **Locks**:

Product	Revenue	Days To Ship
Xtreme Titan Lock	$1,171	1.4
Guardian XL "U" Lock	$1,104	4.4
Guardian Mini Lock	$826	3.0
Guardian ATB Lock	$783	1.7
Guardian "U" Lock	$764	3.1
Xtreme Rhino Lock	$1,306	2.6

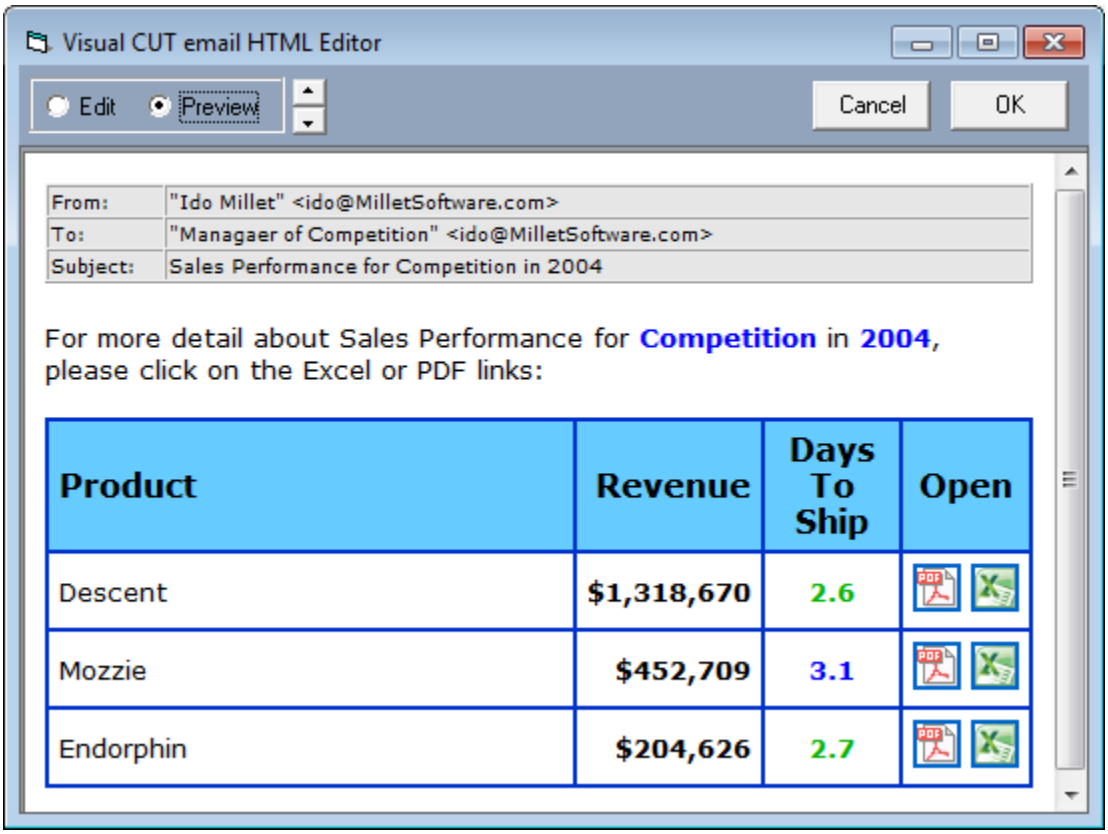
Embedding Hyperlinks to Reports/Files inside HTML Email Messages

Instead of attaching reports to email messages, you can use the email message to provide hyperlinks to the report files located on shared or web folders.

Visual CUT would export the reports directly to shared folders or upload them to web folders using **FTP_Upload** or **SFTP_Upload** arguments. Then, instead of attaching the files to the outgoing email message, you can use Visual CUT's HTML email message editor to embed a hyperlink to the report file. The hyperlink can be simple text or an icon of your choice.

Here is an email example of combining this approach with the HTML table technique described in the previous section. The pdf and excel icons act as hyperlinks for opening report files

.



If you'd like me to send you a sample report demonstrating how Crystal formulas can dynamically construct the hyperlinks and the references to the icons, please send me an email request. I can also send you the pdf and excel icons used in this example.

Using Cascading Style Sheets (CSS) in HTML messages

Just like any other HTML document, you can use CSS directives by embedding them in the header section of the HTML document.

For example, assume you are embedding dynamic HTML tables inside the email message (see previous section for a detailed description of that technique. You may wish to control the font type and font size of table element (identified as **td** in HTML).

Here is an example of using CSS to achieve this:

```
<html>
<head>
        <title>Reseller Balance Report for (@Product_Param}</title>
        <meta http-equiv="content-type" content="text/html; charset=ISO-8859-1">

        <style type='text/css'>
        td{font-family:Verdana; font-size:8pt;}
        </style>
</head>

<body>
      <font face="Verdana, Arial, Helvetica, sans-serif" size="2">
      Hi {Reseller.First_Name},<br><br>
          This message shows your <b>{@Product_Param} </b>purchases, transfers,
          and balance as of {@Nice_Date}. <br>  <br>

          <hr width="100%" size="2">
          <b><big>Your Purchases:</big></b><br><br>
            {@HTML_Table_Sub}<br><br>

          <hr width="100%" size="2">
          <b><big>License Transfers and Balance:</big></b><br><br>
           {@HTML_Table_3_GF1}<br><br>
</font>
</body>
</html>
```

Attaching Multiple Files

To attach multiple files to a single email message, simply specify multiple files and separate them with a **semi-colon (;) Without any spaces.**

If all files are under the same folder, it's enough to specify the full path only for the 1st file.
For example: **c:\temp\Sales.pdf;Returns.pdf**

You can also specify file names using **wild cards**.
For example, in a bursting scenario, to send all pdf files in the current month folder under the current customer:
C:\VC_Exports\{customer.customer_id}\{@Month}*.pdf
and to send all files that start with the current Customer ID:
C:\VC_Exports\{customer.customer_id}*.*

Attaching Optional Files

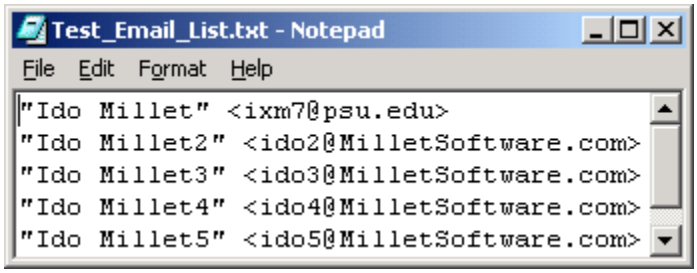
You can indicate that an email attachment is optional by prefixing **<Opt>** to the file path & name. For example, if you need to email Sales.pdf (always) and Returns.pdf (if it exists), use:
c:\temp\Sales.pdf;<Opt>c:\temp\Returns.pdf
- or -
c:\temp\Sales.pdf;<Opt>Returns.pdf

Specifying Multiple (Simple/Composite) Email Addresses

You specify multiple email address destinations in the **To**, **CC**, or **BCC** emailing options, by separating them with a **semi-colon (;) Without any spaces.**
You can also specify **composite** (display name as well as address) email destinations.
For example: **"Ido Millet" <ido@MilletSoftware.com>;"Jane Doe" <Jane@aol.com>**

Specifying Email Distribution Lists in Text Files

To facilitate emailing to a long list of recipients, you can specify in the **To**, **CC**, and **BCC** emailing options a **file name containing a distribution list**.
For example, using Notepad you can create a file such as:



You then specify that Visual CUT should use that file as a distribution list by entering
File:c:\temp\test_email_list.txt
in the **To**, **CC**, or **BCC** emailing options.

Notes:

- you use the word "**File:**" followed by the path & name of the text file containing the email addresses. **Each address should be on a separate line** and it can be specified as an **email address only or as a composite** (display name and address) as shown in the example above.
- As always, you can include references to dynamic fields and formulas. For example, File:c:\distribution_lists\Department_**{@DeptID}**.txt
  would allow you to burst a report by department and use a different distribution list for each department.

Specifying Email Distribution Lists in SQL Queries

In cases where you wish to dynamically retrieve the list of email addresses using an SQL query against an ODBC data source, you can specify in the **To**, **CC**, and/or **BCC** emailing options an expression such as:

MS Access Example:

```
ODBC:Customers::User_ID::Password::SELECT [email] FROM
[Contacts] WHERE [Customer_ID] = '{Customer_ID}'
```

SQL Server Example:

ODBC:CONTACTS::sa::xxxxx::SELECT DISTINCT AHD.ctct.c_email_addr FROM AHD.ctct where AHD.ctct.c_email_addr IS NOT NULL

The expression starts with **ODBC:** followed by 4 elements separated by **::**

1. **ODBC DSN** (Note: could be different from the DSN used for the Crystal report)
2. **User ID** (use a single space if not needed)
3. **Password** (use a single space if not needed)
4. **SQL Statement**

Notes:

- If the SQL statement returns multiple rows, Visual CUT takes care of concatenating the email addresses from all the rows to a single string with semi-colon as the delimiter.
- The SQL statement syntax depends on your database. For example, as shown in the samples above, MS Access uses [ ] around field/table names but SQL Server doesn't.
- You can embed dynamic fields/formula values anywhere inside the expression.
  In the MS Access example, the list of contacts for the Customer in the current bursting cycle would be retrieved. This offers powerful email distribution management options...

Specifying Bursting email Addresses in a Formula

In bursting scenarios, if the database doesn't contain an email address for each Group Level 1, you can use a suppressed Crystal formula (in GH1 or GF1 sections) to return a different email destination for each Group. For example:

```
Select {Branch_Manager.ID}
   Case 101:
      "Jim@Acme.com"
   Case 103:
      "Amy@Acme.com;Don@Acme.com"
   Case 204:
      "Veronique@Acme.com"
   Default:
      "serge@Acme.com";
```

Specifying an Email Reconnect Option for Email Bursting

During Email bursting, the default behavior is to disconnect from the email server after each message. This avoids scenarios where the SMTP server imposes a limit on how many messages can be sent within a single connection. If you know your server doesn't impose such a limitation, or if the number of emails you are sending during a burst operation is always below that limit, you can set an **Email_SMTP_Disconnect_After_Send** option (in DataLink_Viewer.ini) to **FALSE**.

Specifying a Different Character Set

In order to override the default character set (iso-8859-1) you should add an entry to the [Options] section of DataLink_Viewer.ini. For example, in order to support Chinese characters, you should add the following entry:

Email_Char_Set=big5

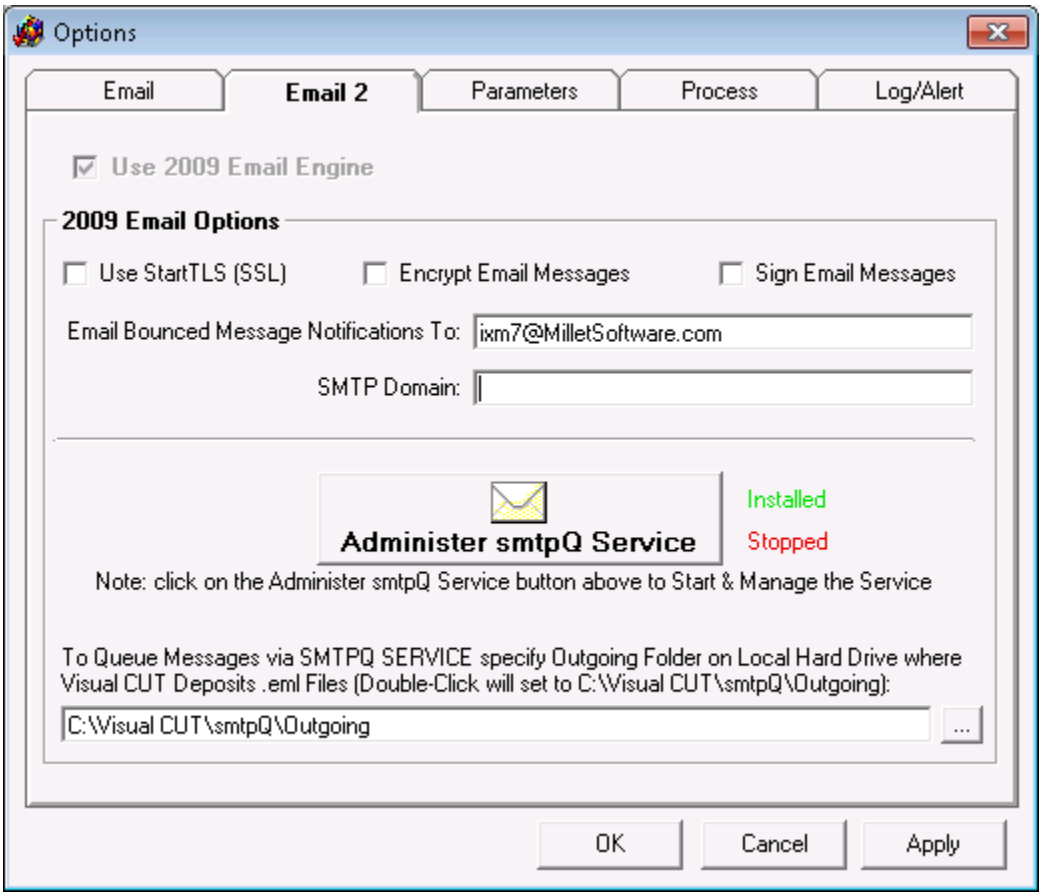
Skipping Emails (VC_Skip_Email)

If you wish to skip the sending of an email message, you can include the text ***VC_Skip_Email*** anywhere inside the Email_To option. For example, you can create a formula that, depending on customer or order status skips the email messages to certain customers. In those cases, instead of returning the customer's email address, the formula would return "VC_Skip_Email". You can drag that formula into the Email_To option and Visual CUT would then skip the emails for those cases.

The advantage of this approach, compared to filtering the report, is that you may want the skip just the emailing while keeping exporting or printing functionality.

New Email Engine

Starting 2011, extra options are available via the 'Email 2' tab in the Options dialog:



Key Features

With the new email engine, when sending messages to multiple recipients, **if the SMTP server rejects one of the recipients as a bad email address the process still continues emailing to the other recipients**.

You can also use secure connections to the SMTP server (NTLM, CRAM-MD5, SSL, STARTTLS). The **StartTLS** option allows you to communicate securely with SMTP servers that require that option. Note: use the **Email_StartTLS** command line argument ("True", "False", or dynamic reference) to override the default specified in the Options dialog.

The **Encrypt Email Messages** option allows you to encrypt the message so that only the recipient can open it. Visual CUT automatically searches for and uses the public key found in the first non-expired certificate matching the recipient's email address (the recipient will be using their private key). The search for matching key includes the Current User's personal certificate store and then the Local Machine's personal certificate store. Use the **Email_Send_Encrypted** command line argument ("True", "False", or dynamic reference) to override the default specified in the Options dialog.

The **Sign Email Messages** option allows you to sign (using your private key) the message so that the recipient can verify (using your public key) that a) you are the true sender of the message, and b) the message content hasn't been altered. Note: use the **Email_Send_Signed** command line argument ("True", "False", or dynamic reference) to override the default specified in the Options dialog.

The **Email Bounced Message Notification To** option allow you to **specify a destination for bounced messages**. Note: use the Note: use the **Email_Bounce_Address** command line argument (static or dynamic reference) to override the default specified in the Options dialog.

The **SMTP Domain** option allow you to **specify a domain in cases where the SMTP server used integrated authentication**. Note: use the **Email_SMTP_Domain** command line argument (static or dynamic reference) to override the default specified in the Options dialog.

Queuing Emails & The smtpQ Service

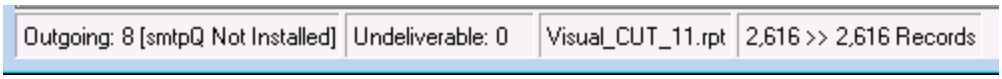
With the new email engine, you can queue outgoing email messages in an 'Outgoing' folder monitored by a new smtpQ service. If you enter any path in the **Outgoing Folder** option in the 'Email 2' tab, Visual CUT would use that approach. You can also activate this behavior by using the command line argument of "Email_Outgoing_Folder:Path_to_Some_Folder".

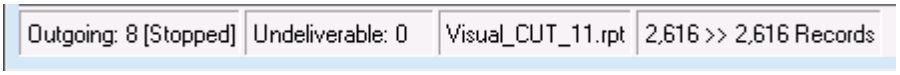
A special **smtpQ Service** (a separate process that always runs in the background) is responsible for handling the email messages queued in the '**Outgoing**' folder. You can install, start/stop, and administer the smtpQ Service by clicking on the 'Administer smtpQ Service' button or by double-clicking the status labels on the right:

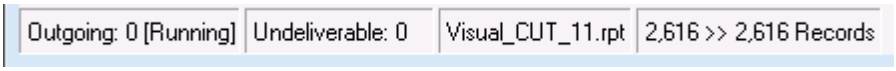
Note: you need to have **Administrator rights** to manage the smtpQ service. Also, I've seen one case where the user had to lower the Windows '***User Access Control***' security settings before the machine allowed Visual CUT to manage the smtpQ service.

Monitoring Email Queuing

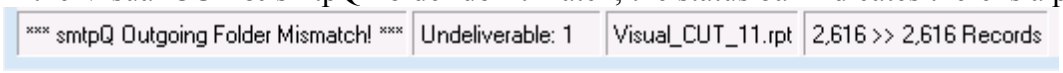
If an Outgoing Folder is specified, Visual CUT shows the **number of email messages in the Outgoing and Undeliverable folders** in the first 2 panels of the status bar.
The panel showing the number of messages in the Outgoing Folder, also indicates if the smtpQ service is **Not Installed**, **Stopped**, or **Running**. The information is refreshed every 5 seconds:







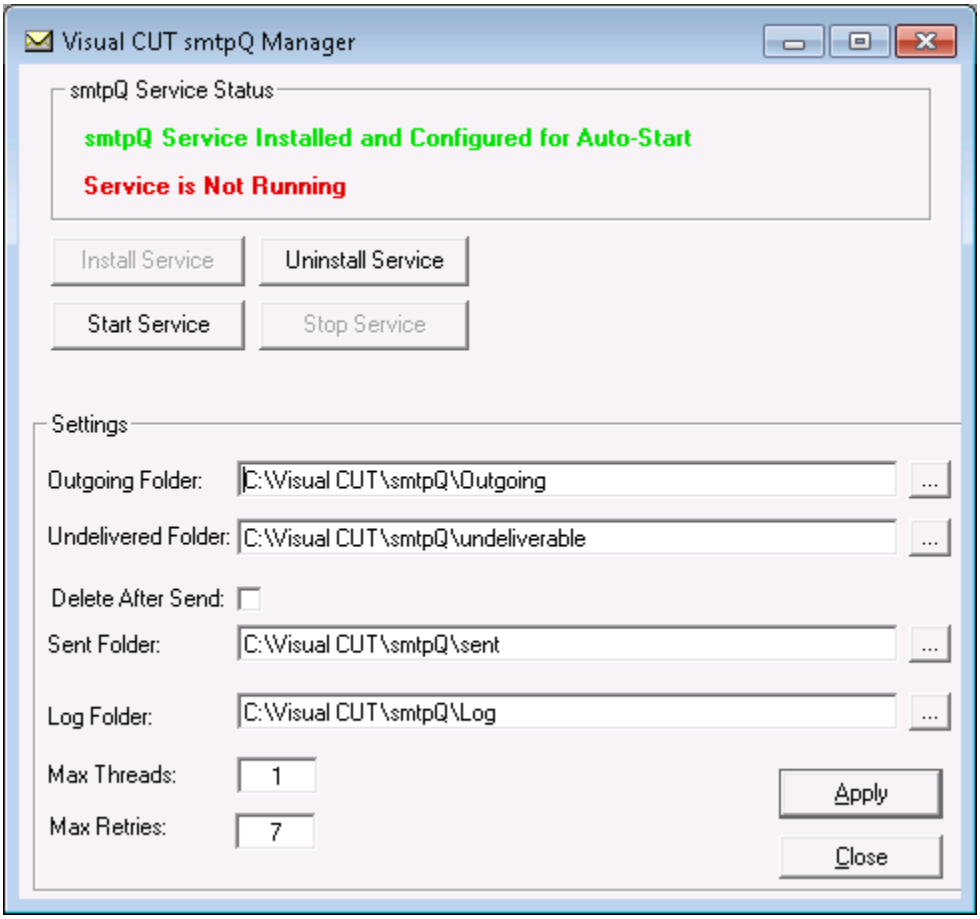
If the Visual CUT & smtpQ Folder don't match, the status bar indicates there is a problem:



As an added convenience, you can open the **Outgoing** folder or the **Undeliverable** folder in File Explorer by double-clicking their panel.

smtpQ Administration

Clicking on the Administer smtpQ Service button brings up the following window:



The Install Service button installs the service so that each time your computer starts the service will start automatically. Once the service is installed, you can Start or Stop the service using the appropriate buttons. You can also stop and start the service from the Windows Control Panel, Administrative Tools, Services.. (it will show up as ***Chilkat SMTPQ***):

Important: while the ***Outgoing Folder*** option in Visual CUT controls where Visual CUT deposits outgoing email messages (as .eml files), the ***Outgoing Folder*** option in the smtpQ Service controls what folder is monitored by the service. Obviously, these two options should point at the same folder if you wish to actually send emails automatically.

Keep the **Max Threads** option low unless you are sure the SMTP server would allow you to send a high volume of emails through many concurrent sessions.

The **Max Retries** option allows you to specify how many attempts would be made to send a failing email message (for example, if connectivity to your SMTP Server was lost). The delays before each retry keep increasing from 1 to 10th {5 sec, 10 sec, 15 sec, 1 min, 1.5, 2, 5, 10, 15, 20 minutes}.

As soon as a new .eml file appears in the ***Outgoing Folder***, the smtpQ Service will attempt to send it. If it succeeds, it will either move the file to a ***Sent folder*** or you can elect to have successful messages deleted. If after several minutes of repeated retries, the service fails to send the message, it will move it to an ***Undeliverable folder***.

These email messages are deposited as **.eml files that can be opened in Windows Mail or Outlook Express and manually transferred between folders**. For example, if you copy .eml files from the 'Sent' folder to the 'Outgoing' folder, they will be emailed again. This can be useful in cases where the receiver junked your original message by mistake. Among other benefits, **this allows you to archive outgoing emails and easily recover from email service disruptions**. For example, your email server may be down or your email service provider may not allow more than 100 messages per hour.

After making changes to the options in the smtpQ Service Manager dialog, you should click the Apply button, and then stop and restart the service to have these changes take effect.

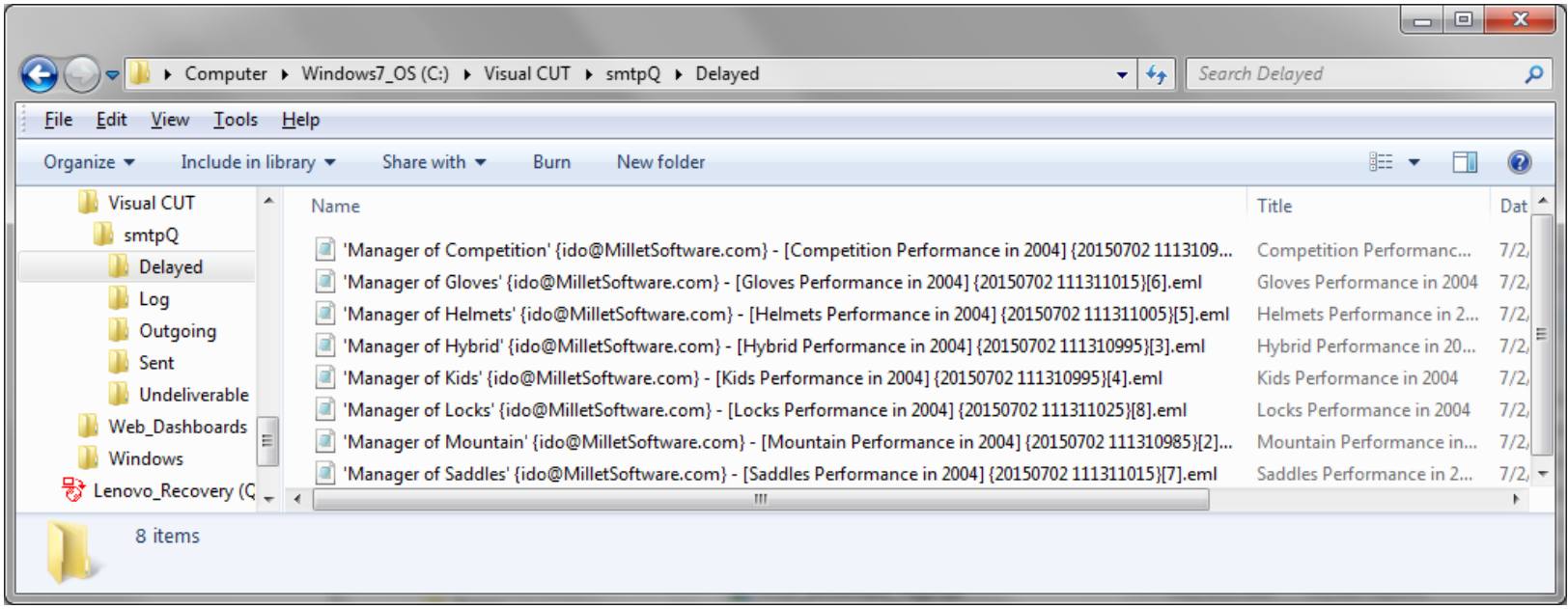
Naming Scheme of .eml Files

To facilitate the process of locating specific messages, the .eml files are named using three parts:

a) the name and address of the first email **recipient**
b) [the **Subject** of the email message]
c) {the **date and time** the message was generated}
d) [The **group number** being processed]

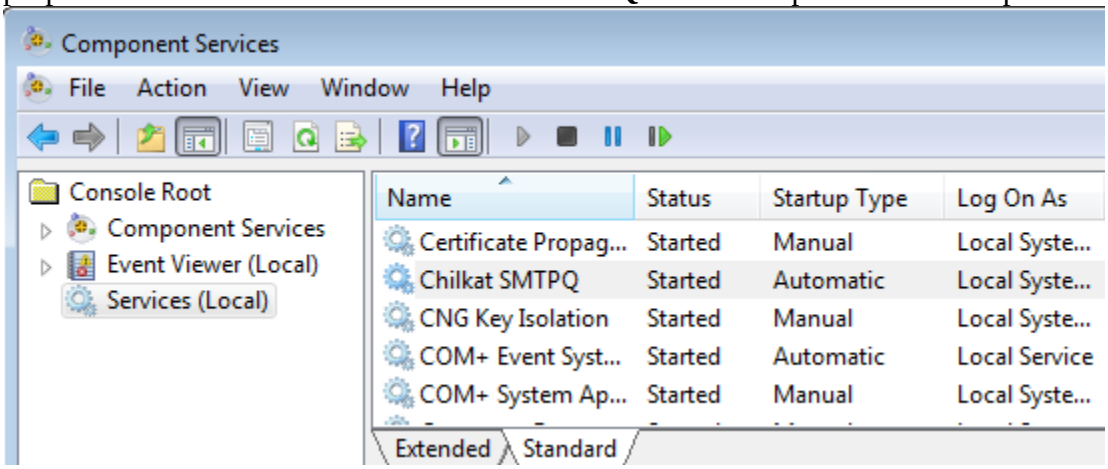
Remember that these .eml files can be opened using software such as Windows Live Mail, Outlook Express, etc..

Here's an example:

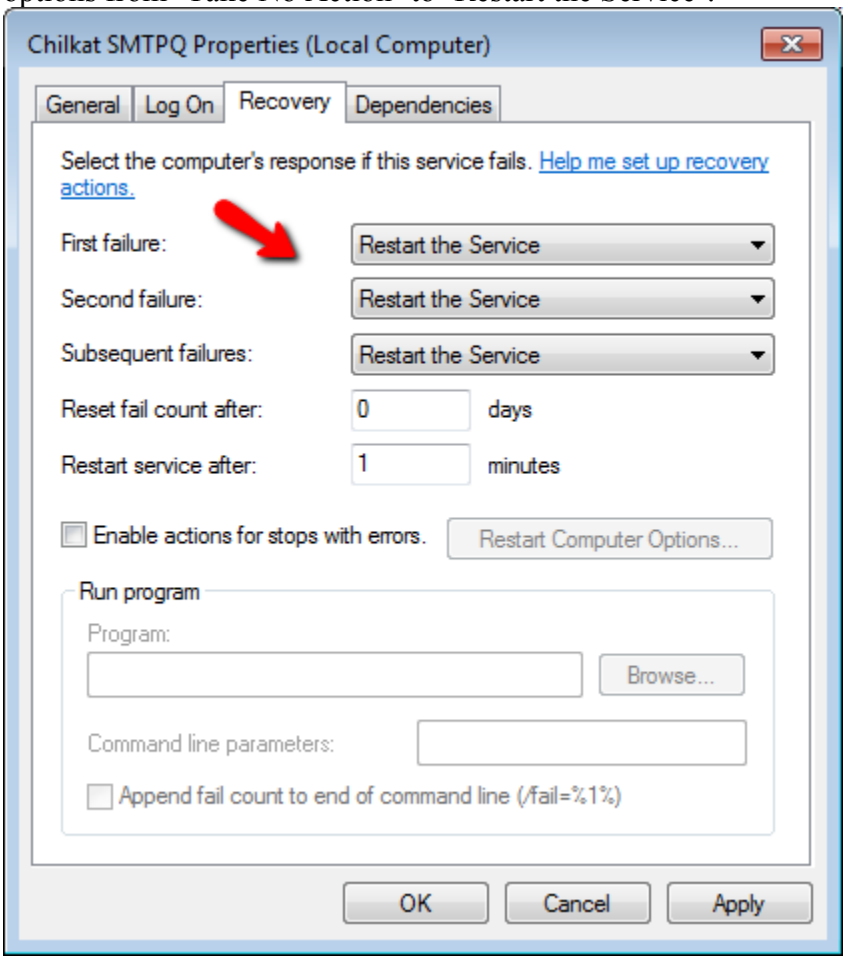

Computer ▸ Windows7_OS (C:) ▸ Visual CUT ▸ smtpQ ▸ Delayed
Search Delayed
File Edit View Tools Help
Organize Include in library Share with Burn New folder
Visual CUT
smtpQ
Delayed
Log
Outgoing
Sent
Undeliverable
Web_Dashboards
Windows
Lenovo_Recovery (C
Name
Title
'Manager of Competition' {ido@MilletSoftware.com} - [Competition Performance in 2004] {20150702 1113109...
Competition Performanc...
'Manager of Gloves' {ido@MilletSoftware.com} - [Gloves Performance in 2004] {20150702 111311015}[6].eml
Gloves Performance in 2004
'Manager of Helmets' {ido@MilletSoftware.com} - [Helmets Performance in 2004] {20150702 111311005}[5].eml
Helmets Performance in 2...
'Manager of Hybrid' {ido@MilletSoftware.com} - [Hybrid Performance in 2004] {20150702 111310995}[3].eml
Hybrid Performance in 20...
'Manager of Kids' {ido@MilletSoftware.com} - [Kids Performance in 2004] {20150702 111310995}[4].eml
Kids Performance in 2004
'Manager of Locks' {ido@MilletSoftware.com} - [Locks Performance in 2004] {20150702 111311025}[8].eml
Locks Performance in 2004
'Manager of Mountain' {ido@MilletSoftware.com} - [Mountain Performance in 2004] {20150702 111310985}[2]...
Mountain Performance in...
'Manager of Saddles' {ido@MilletSoftware.com} - [Saddles Performance in 2004] {20150702 111311015}[7].eml
Saddles Performance in 2...
8 items

smtpQ Service Failure Action Properties

Each time the smtpQ service is installed or reinstalled, it is a good idea to set Recovery properties in case it fails. Locate ***Chilkat SMTPQ*** in the Component Services panel in Windows:


Component Services
File Action View Window Help
Console Root
Component Services
Event Viewer (Local)
Services (Local)
Name Status Startup Type Log On As
Certificate Propag... Started Manual Local Syste...
Chilkat SMTPQ Started Automatic Local Syste...
CNG Key Isolation Started Manual Local Syste...
COM+ Event Syst... Started Automatic Local Service
COM+ System Ap... Started Manual Local Syste...
Extended Standard

Right-click the service, select 'Properties' and go to the Recovery tab. Change the failure options from 'Take No Action' to 'Restart the Service':


Chilkat SMTPQ Properties (Local Computer)
General Log On Recovery Dependencies
Select the computer's response if this service fails. Help me set up recovery actions.
First failure: Restart the Service
Second failure: Restart the Service
Subsequent failures: Restart the Service
Reset fail count after: 0 days
Restart service after: 1 minutes
Enable actions for stops with errors.
Restart Computer Options...
Run program
Program:
Browse...
Command line parameters:
Append fail count to end of command line (/fail=%1%)
OK Cancel Apply

Slowing Down Outgoing Emails

If your SMTP service provider imposes a limit on how many emails you are allowed to send within a given time period, you may need to slow down the email queuing process:

Set the Visual CUT Outgoing folder as: c:\Visual CUT\smtpQ**Delayed**
and the Outgoing folder monitored by smtpQ as: c:\Visual CUT\smtpQ**Outgoing**
This means Visual CUT would deposit eml files in the **Delayed** folder, so they won't be sent unless they get moved to the smtpQ **Outgoing** folder.

The following batch file would move the files from the **Delayed** folder to the **Outgoing** folder at a rate of 1 per **10** seconds.

```
---------------------------------------
@ECHO OFF
for %%i in ("c:\Visual CUT\smtpQ\Delayed\*.eml") do (
Move "%%i" "c:\Visual CUT\smtpQ\Outgoing\"
timeout /t 10 /nobreak
)
---------------------------------------
```

A timeout of **10** seconds would result in only 6 emails per minute. If you want to increase the emailing rate to 30 per minute, reduce the timeout value to **2**.
When the process runs out of files, the batch file simply goes away.

You can a) trigger the batch file manually, b) schedule it to run every N minutes, or
c) call it in Visual CUT using the following command line argument:
"**After_Success_Batch:Whole>>c:\Batch\Move_emls_ from_Delayed_to_Outgoing.bat**"

To avoid concurrent running instances of the batch file, here is a version that writes a **FilesBeingMoved** text file at the start of the process and deletes it at the end.
When the batch starts, it aborts if the file already exists.

```
---------------------------------------
IF EXIST "c:\Visual CUT\smtpQ\Delayed\FilesBeingMoved.txt" (
REM Do nothing
) ELSE (
  @ECHO OFF
  REM Create lock file
  ECHO > "c:\Visual CUT\smtpQ\Delayed\FilesBeingMoved.txt"
  for %%i in ("c:\Visual CUT\smtpQ\Delayed\*.eml") do (
  Move "%%i" "c:\Visual CUT\smtpQ\Outgoing\"
  timeout /t 10 /nobreak
  )
REM Delete the lock file afterwards
DEL "c:\Visual CUT\smtpQ\Delayed\FilesBeingMoved.txt"
)
---------------------------------------
```

Generating Twitter or SMS Messages via Email

You can send **SMS** messages from Visual CUT using email. You just need to know the **number** and the **carrier** of the recipient. See detail at http://www.makeuseof.com/tag/email-to-sms/

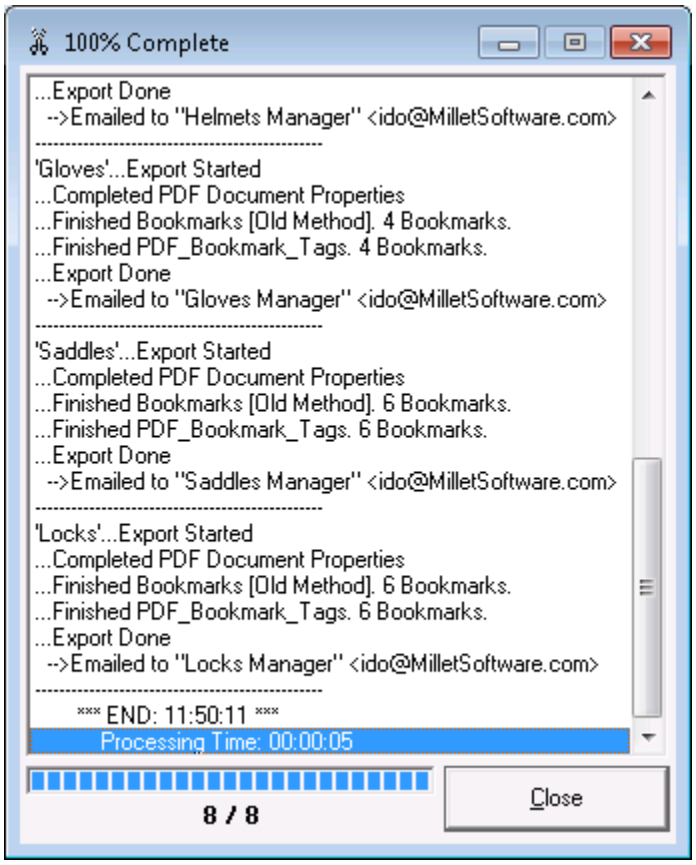
In order to send **tweets** via Visual CUT, you can send email messages via a service that acts as a bridge to Twitter. The text below was provided by a Systems Analyst at a Fire Department that asked to remain nameless:

> Our Fire Department uses Visual CUT to send various tweets via Twitter based on specific call for service data. Visual CUT is set up via a scheduled task to run every minute and looks for any newly-created records qualifying for tweeting. If a qualified record is found, Visual CUT sends an email to a specified email address that automatically posts to twitter. This is a free service provided by Twitter Counter (http://twittercounter.com/pages/twittermail/).
>
> As part of the same process, Visual CUT also exports some data fields via ODBC to another table. This keeps track of which incidents were tweeted about. The same process also uses that table to determine if a previously tweeted incident has been cleared. This allows us to also use Visual CUT to send an all clear Twitter message.
>
> We don't tweet out every incident, as many of our incidents are medical. We use a separate Visual CUT process at the end each month to send a count of the total number of incidents. This way our residents are aware of a total volume as opposed to just a couple of calls a day that get tweeted about.

Tweets
15m
has cleared incident #12012579 in the 100 Block of OCEAN PARK BLVD.
Expand
18m
responding to a Traffic Collision in the 100 Block of OCEAN PARK BLVD. Possible traffic congestion. Inc.#12012579
Expand

‘Start Process’ button and Progress Window

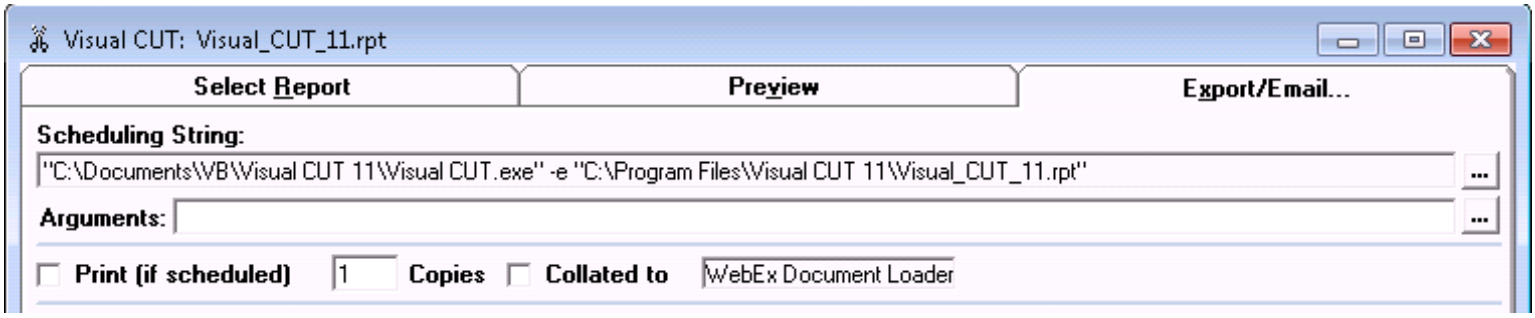
This button simply starts the process of Bursting, Exporting, and Emailing as specified by the options above. During the process, a window traces the progress of the operation:



While in progress, instead of a **Close** button, you will see a **STOP** button allowing you to abort the operation.

Tip: at the end of a process, if you double-click the export file name field
(in the export/email tab) , Visual CUT will open that folder in file explorer for you.

Scheduling Area



Scheduling String

The scheduling string text provides the necessary command line to trigger Visual CUT processing for this report. The button to the right allows you to select or create a batch file to hold this command line.

Arguments

The argument field allows you to specify and save extra command line arguments. You should review the table of command line arguments at the end of this user manual. They allow you to tackle many use scenarios that are impossible to address with other tools. By specifying the arguments in this field, you remove the need to specify them in the command line. Also, this allows you to take advantage of the power of these argument even during interactive processing. The button to the right opens a text editor that makes it easier to review/edit the arguments.

Scheduled Printing

This area also shows what printer is currently associated with the report. Visual CUT sends reports scheduled for printing to that printer. Within Crystal, you can associate each report with a different printer using the ***File, Printer Setup…*** dialog. As explained later, you can override the printer destination when scheduling or invoking Visual CUT via command lines.

The **number of copies option can be dynamic** (drag & drop a formula into that option) or static (enter a value). You can also specify the collation option.

Note that these printing options apply only for scheduled (command line) invocations of Visual CUT. if you want the report to be printed during an interactive session, you should simply print it from the Preview tab).

Scheduling

By clicking on the ***Save*** button, all report processing options, parameters, and logon information (encrypted) are saved for later use by ***Visual CUT***. This allows the application to process any report in ***unattended mode*** by simply invoking the executable (***Visual CUT.exe***) with a proper command line (using a *.Bat file, a RUN command, a desktop shortcut, or a call from any other application).

The recommended, and most typical, approach is to
1) place the command line in a batch (.bat) file, and
2) call the batch file from the Windows Task Scheduler

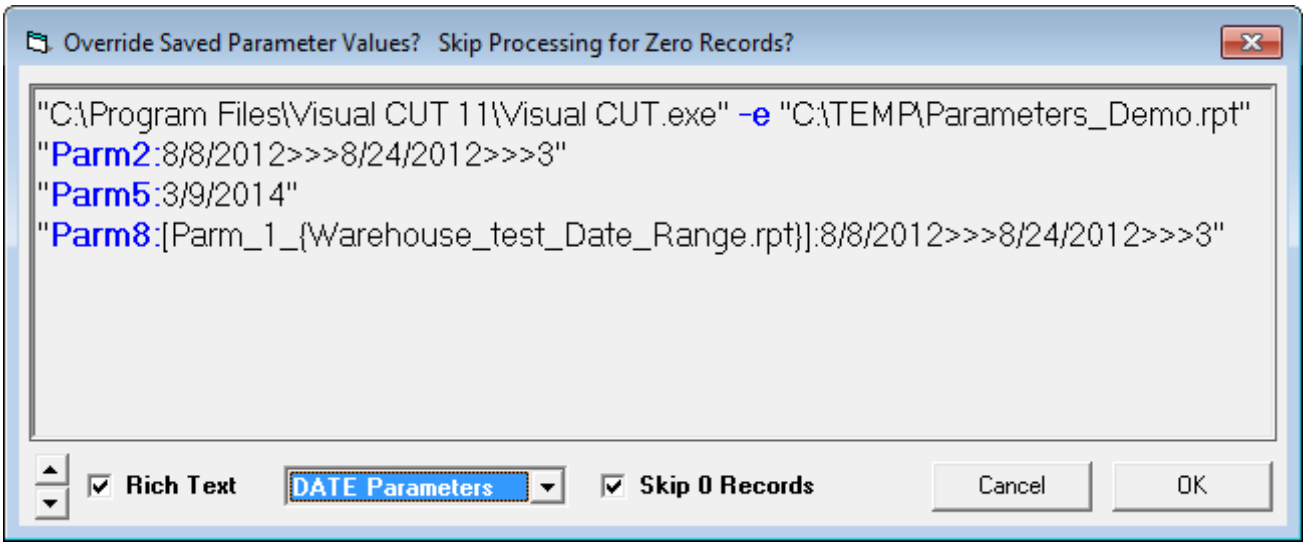
Place the command line in a batch (.bat) file

The "Scheduling String" in Area 5 provides you with the command line needed to run a report via Visual CUT in unattended mode. You may copy and paste that command line into a batch file but a better approach is to click the button to the right of the scheduling string.
This prompts you to enter a new batch file name or select an existing batch file.
The command line is then inserted into the new/existing batch file and you can elect to immediately view/edit the batch file inside Notepad.

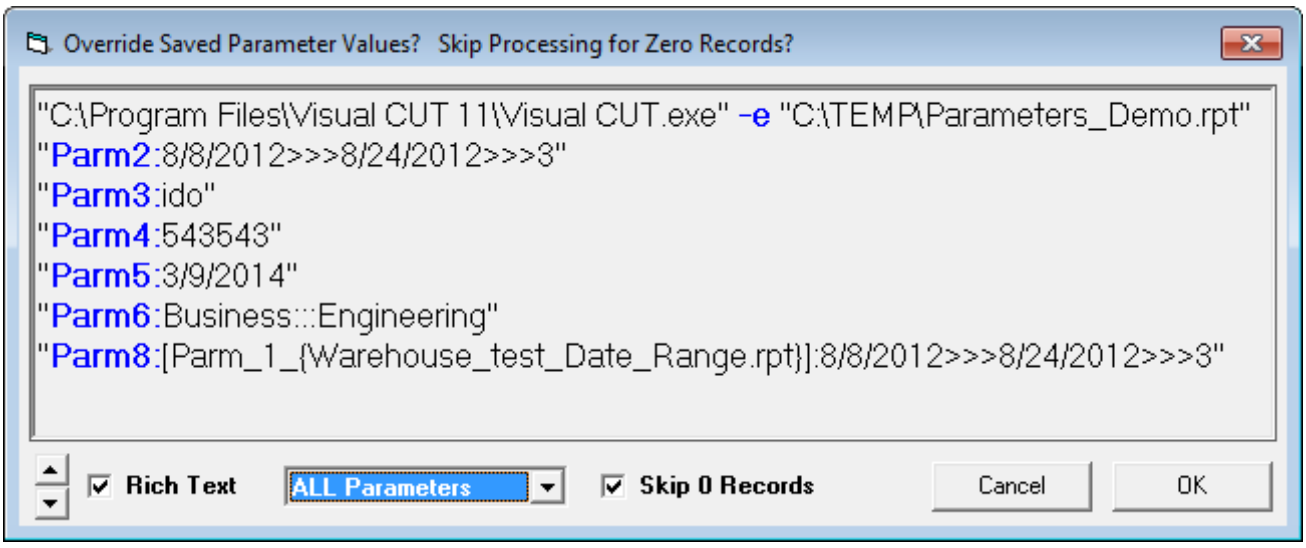
You can test the process by double-clicking the batch file. This should trigger processing of the report in Visual CUT.

Command Line Wizard (Parameters & Skip Zero Records)

If the report has parameters, the ... button to the right of the scheduling string starts a dialog with options to include parameters in the initial command line. If the report has Date or DateTime parameters, the dialog defaults to including them. You can then edit the static values in Notepad, and replace them with **Date Constants** such as Yesterday, Start_Month_Minus_1, etc. (see Date Constants section in the user manual).



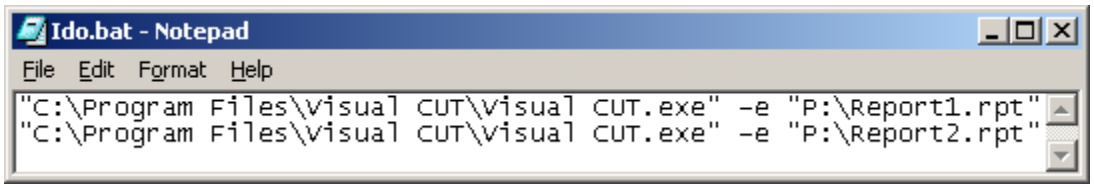
If you select 'ALL parameters' from the drop-down, all active parameters are included.
Note: you should include parameter arguments only if you need to override the parameter values saved when you previewed the report in Visual CUT.
(Parm1 is not included in the case below because it is an unused parameter):



If you uncheck the 'Skip 0 Records' option, the execute flag in the command line changes from **-e** to **-E**. With the default of **-e** Visual CUT skips processing if the main report has zero records (allowing easy implementation of exception reports and email alerts).

Processing of Multiple Reports in a Single Batch (.BAT) File

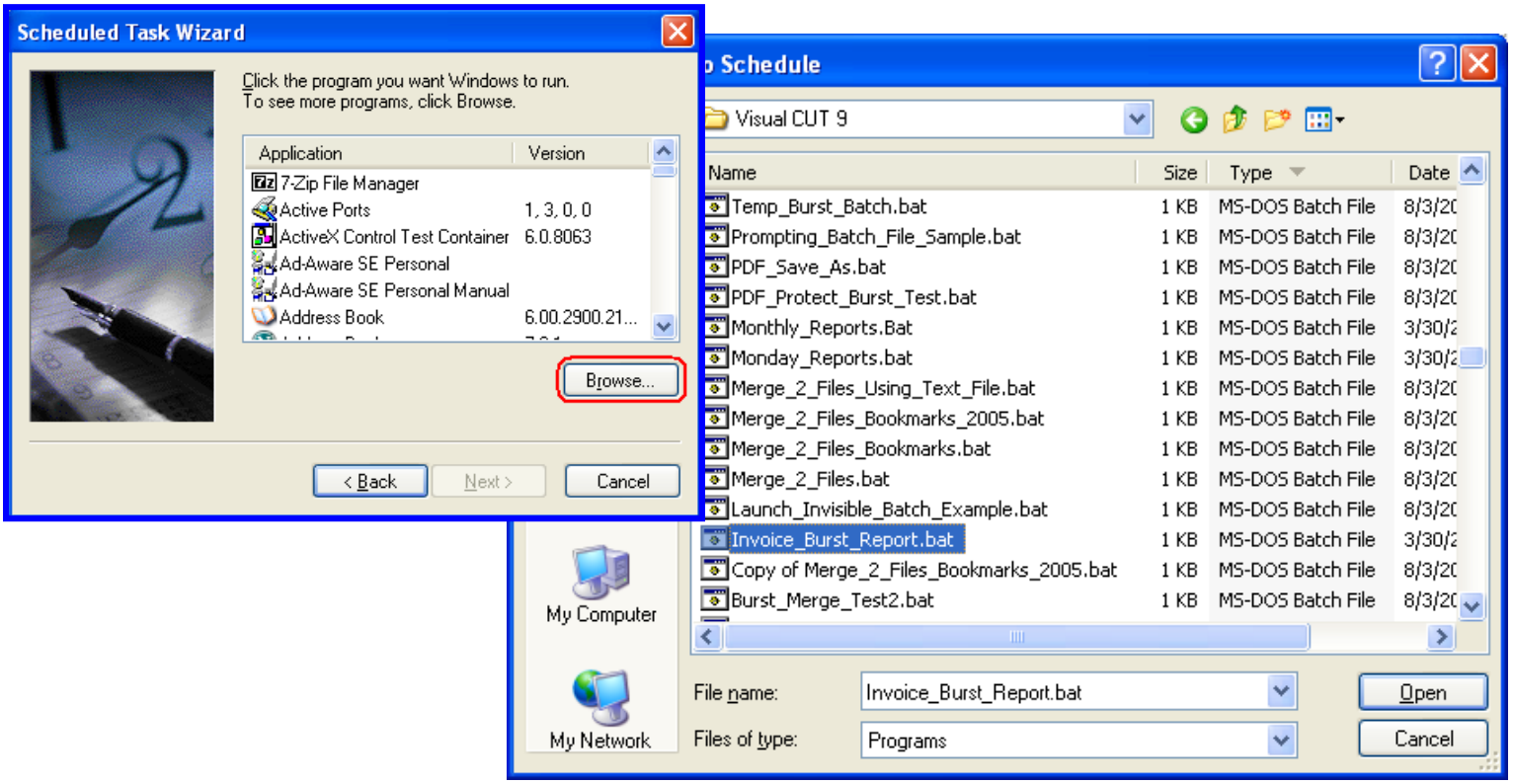
You can use several command lines inside a single batch (.bat) file to trigger processing (manually or via scheduling) of multiple reports. For example, double-clicking (or scheduling) the following **Ido.bat** file would trigger processing, one after the other, of **Report1.rpt** and then **Report2.rpt**:


Ido.bat - Notepad
File Edit Format Help
"C:\Program Files\Visual CUT\Visual CUT.exe" -e "P:\Report1.rpt"
"C:\Program Files\Visual CUT\Visual CUT.exe" -e "P:\Report2.rpt"

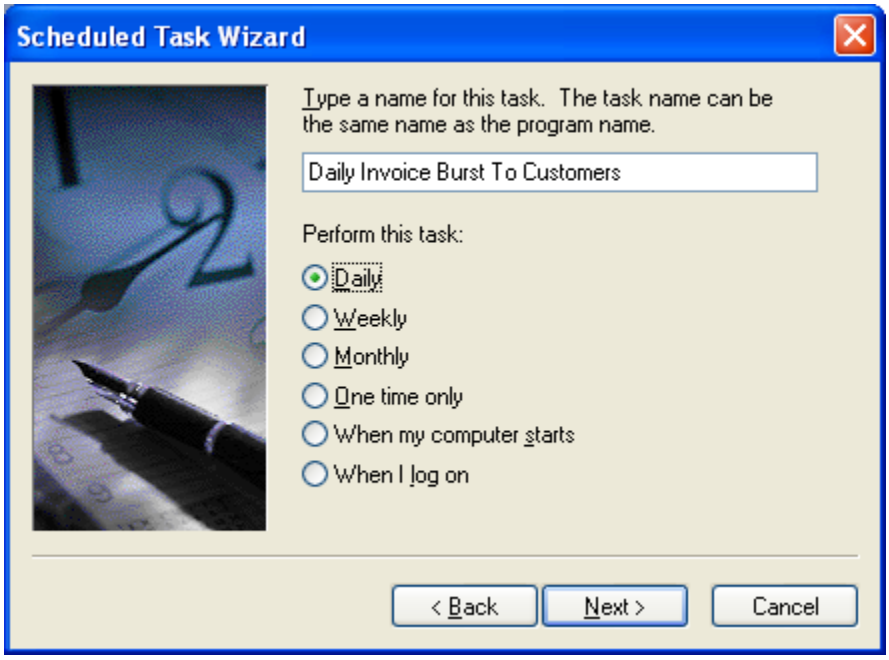
Call the Batch file from Windows Task Scheduler

While there are many commercial schedulers, the free Windows Task Scheduler (already installed as part of your Windows operating system) is capable of handling most scheduling requirements. Here's how you can go about scheduling a batch file:

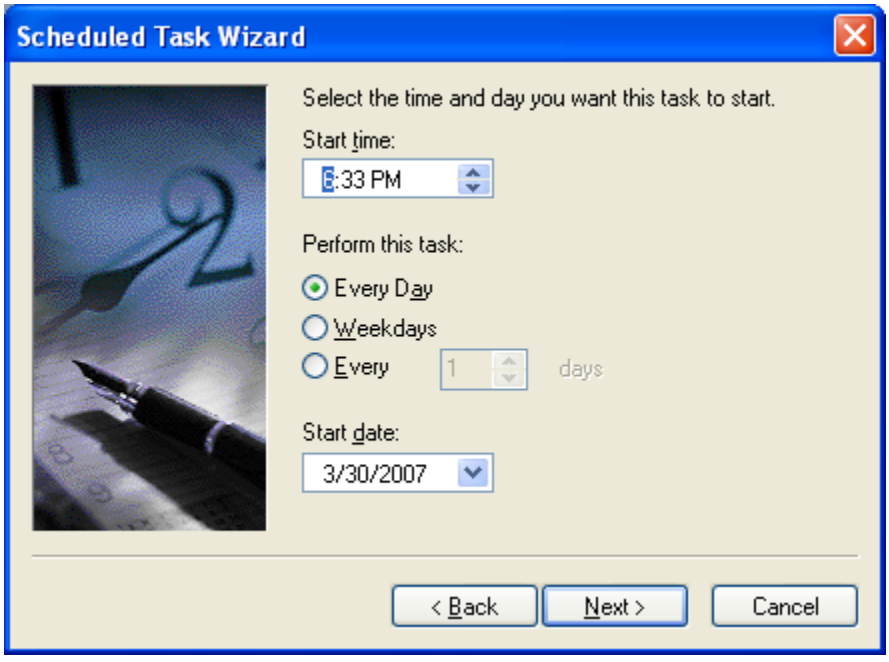
1. In ***Windows***, click on Start, Settings, Control Panel, Scheduled Tasks
2. Double-Click ***Add Scheduled Task***
3. In the Scheduled Task Wizard, click the **Browse** button,
   and **select the Batch file** containing the Visual CUT command line(s).



4. Specify Name and Frequency for the scheduled task:


Scheduled Task Wizard
Type a name for this task. The task name can be the same name as the program name.
Daily Invoice Burst To Customers
Perform this task:
Daily
Weekly
Monthly
One time only
When my computer starts
When I log on
< Back
Next >
Cancel

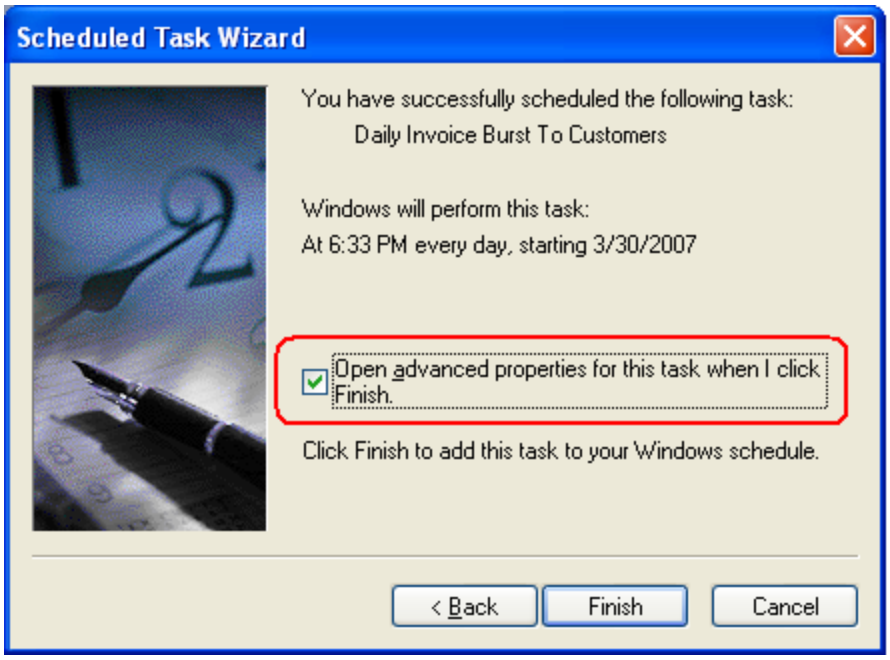
5. Specify Schedule options. These options vary depending on the frequency you selected above.



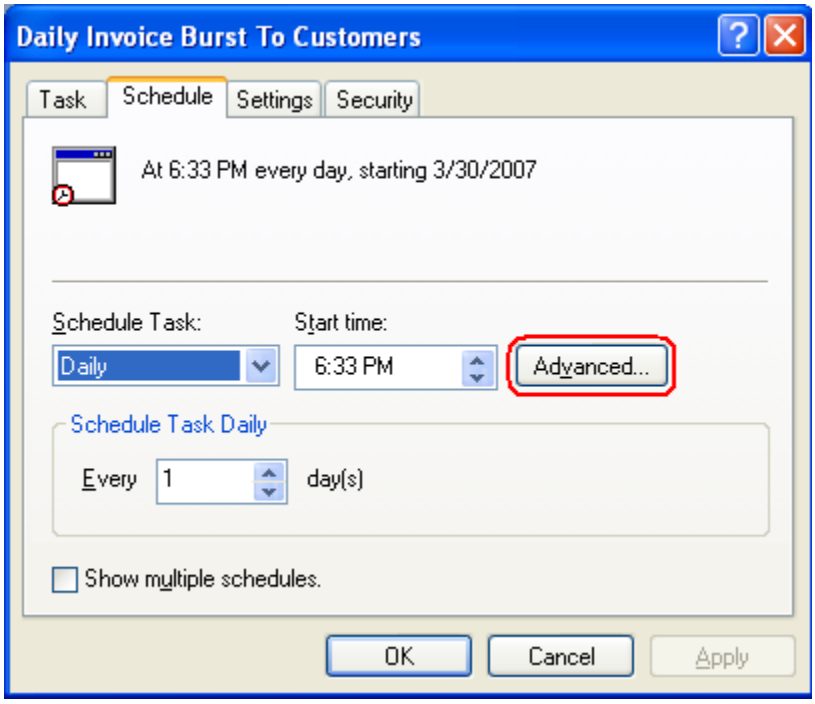
6. Specify your password:

Scheduled Task Wizard
Enter the name and password of a user. The task will run as if it were started by that user.
Enter the user name:
ixm7
Enter the password:
Confirm password:
If a password is not entered, scheduled tasks might not run.
< Back
Next >
Cancel

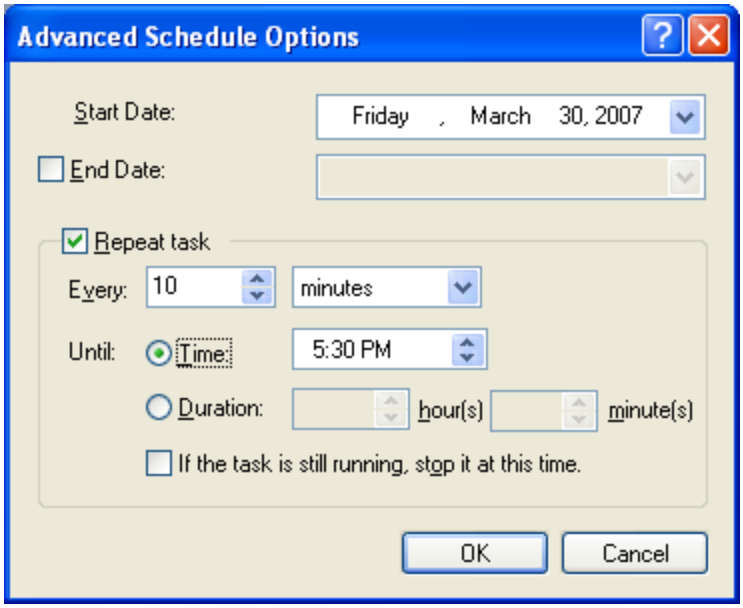
7. Select the option to Open advanced properties and click Finish.


Scheduled Task Wizard
You have successfully scheduled the following task:
Daily Invoice Burst To Customers
Windows will perform this task:
At 6:33 PM every day, starting 3/30/2007
Open advanced properties for this task when I click Finish.
Click Finish to add this task to your Windows schedule.
< Back
Finish
Cancel

8. The **Schedule tab** in the advanced properties dialog provides an **Advanced button**:

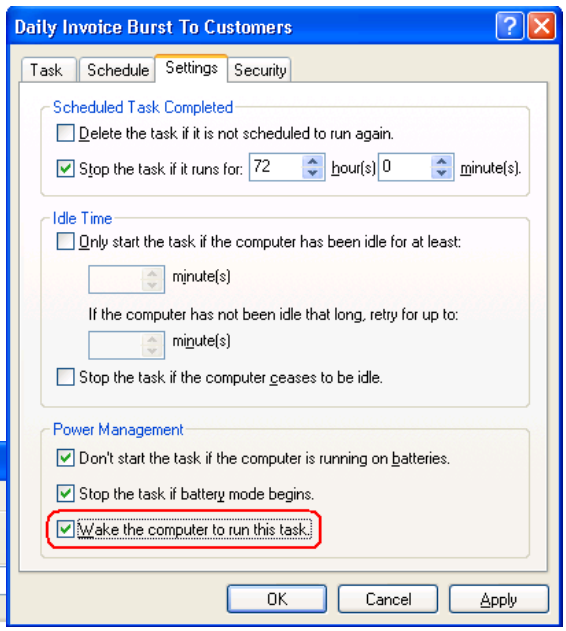


Clicking on that button, provides various advanced scheduling options:

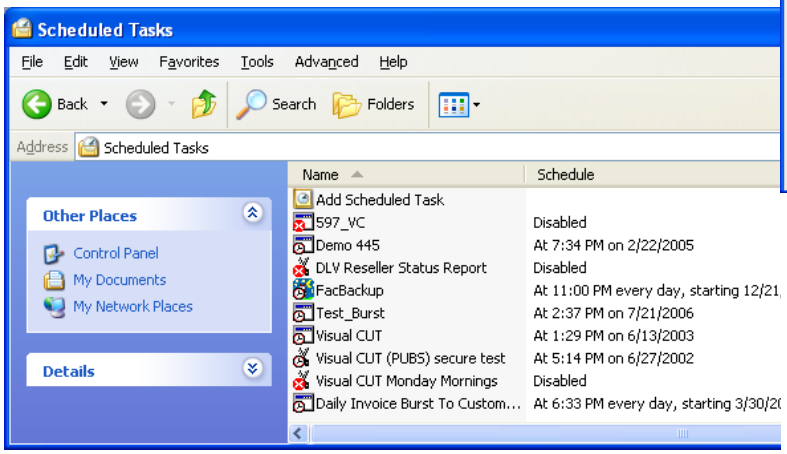


Using the Settings Tab, you can specify other advanced options such as minimum idle time requirement before the scheduled task is allowed to begin.

Be sure to turn on the option of "Wake the computer to run this task."
For a description of why this setting is important (in the context of the Vista operating system), see: http://support.microsoft.com/kb/930133.


Daily Invoice Burst To Customers
Task
Schedule
Settings
Security
Scheduled Task Completed
Delete the task if it is not scheduled to run again.
Stop the task if it runs for: 72 hour(s) 0 minute(s).
Idle Time
Only start the task if the computer has been idle for at least:
minute(s)
If the computer has not been idle that long, retry for up to:
minute(s)
Stop the task if the computer ceases to be idle.
Power Management
Don't start the task if the computer is running on batteries.
Stop the task if battery mode begins.
Wake the computer to run this task.
OK
Cancel
Apply

9. After the task is scheduled, it is listed and tracked under the ***Scheduled Tasks*** folder:



By double- or right-clicking-clicking any task in the Scheduled Tasks folder, you can bring up the scheduling dialog, change settings, rename, run, or disable a task.

Scheduling Issues

Mapped → UNC File Paths

The task scheduler, given that it runs under the local SYSTEM account, may not recognize mapped network drives. Recent versions of Visual CUT automatically convert mapped paths to UNC paths when you open an rpt from a mapped folder, or when you set up export file locations.
For example: \\server123\ixm7\{@Year}\Sales for {Product_Type.Name}.pdf
Instead of: P:\{@Year}\Sales for {Product_Type.Name}.pdf
But if the report uses a mapped drive in the data source setup, you should change the data source definition to use a UNC path instead.

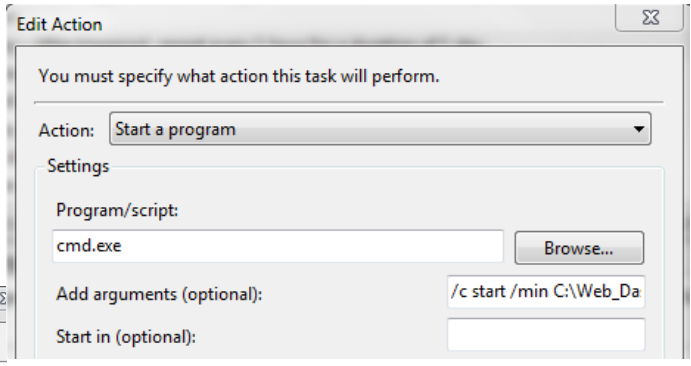
Windows 7/8 Task Scheduler

Turn on the "**Run with highest privileges**" task property ('General' tab).
The user account that is setting up the task should be a local administrator.

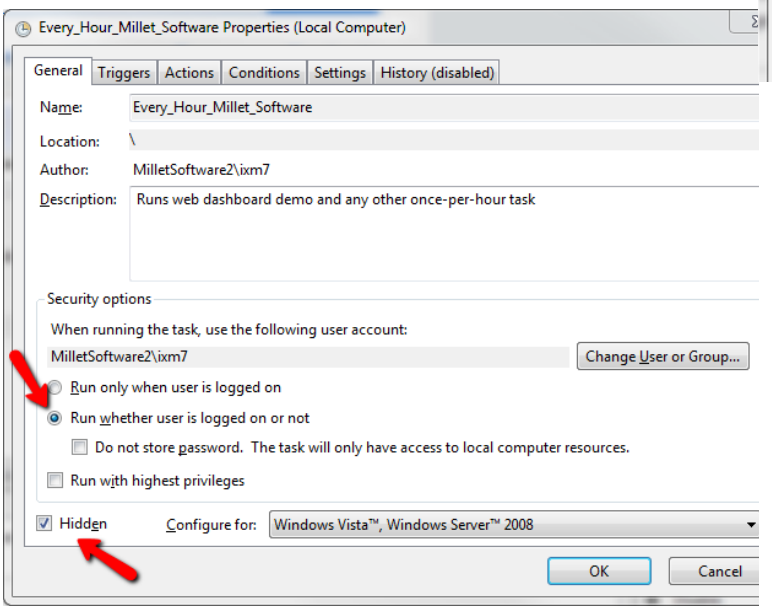
On the actions tab, set action to **Start a program**
Program/script: C:\Windows\System32**cmd.exe** (or just **cmd.exe**)
Add arguments: **/C** C:\Some_Folder\Your_Bat_File.bat

Minimize or Hide the Command Window

To avoid being distracted by the batch file window each time it gets triggered, use the following in the Arguments area (see this for explanation):
/c start /min C:\Some_Folder\Your_Bat_File.bat ^& exit



Better yet, in the General tab, turn on 'Run whether user is logged on or not' and turn on the Hidden checkbox.



Control Processing Options When Encountering Zero Records

The command line is constructed as the name and path of the executable (surrounded by quotes), followed by **–e** or ***-E*** and the path and file name of the report (surrounded by quotes):

"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt"

Specifying **–e** or ***–E*** in the string above causes automatic "***Execution***" based on encrypted and stored logon information, stored parameter values, and the various processing options specified & saved during an interactive session in ***Visual CUT.***

–e aborts processing of empty reports while ***–E*** doesn't.

Specifying **–s** in the string above causes automatic "***Show***" (Preview Tab) based on logon and parameters specified & saved during a previous session in ***Visual CUT.***
You can also use this to create a simple **shortcut** for quick viewing of a report.

If no processing option (-e, -E, or –s) is specified in the Scheduling string, the application's Preview Tab would open for the specified report prompting for Parameters and Logon information if required.

E-mailing Alert Messages and Exception Reports

Imagine managers at your company wish to receive daily reports showing exceptions such as orders with very low profit margins, products with very low inventory levels, late shipments, or production runs with very high scrap rates. In most days these exception reports should be empty and should not reach the managers.

Using Visual CUT you can schedule these reports to run as frequently as you wish, but with **–e** rather than **–E** in the command line. This would ensure that during each scheduled processing cycle, the e-mail message, and (optionally) the attached report would be sent only if exceptions were indeed found. If no records pass the record selection criteria, exporting, e-mailing, and printing of the report are simply aborted.

This logic applies to the main report. A case of no records in the main report but some records in a subreport is still considered as no records.

Note: to avoid repeating the same exception email, see ***Avoiding Duplicate Processing***.

Management By Exceptions and ***Business Activity Monitoring*** (BAM) are powerful approaches to management, and ***Visual CUT*** makes them easy… ☺

Using Command Line Arguments

Arguments to Specify Parameter Values

When you open a report in Visual CUT, the values you specify in response to parameter prompts are saved in the internal database (Visual CUT.mdb). In some cases, you may want to invoke Visual CUT processing via command line calls while overriding some or all of these stored parameter values. You can do this by specifying additional command line arguments.

So far, the string used to schedule or launch processing of reports was described as:
"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt"
or
"C:\Program Files\Visual CUT\Visual CUT.exe" -E "C:\Program Files\Visual CUT\Visual_CUT.rpt"

You can add optional arguments to the command lines, which identify the parameter (by its number) and specify its value. For example, using the following string would override the first parameter value saved by Visual CUT with the value of 1988 (in the sample ***Visual_Cut.rpt*** report, the first parameter is the YEAR).

"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt" "Parm1:1998"

The value of a date parameter would be specified as:
"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt" "Parm1:3/10/2003"

Note that the syntax is constructed as the word "Parm", followed by the number of the parameter (according to the order of parameters shown in Crystal), followed by a colon and the value.

Here's how you would specify the 1st and 3rd parameter values:
"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt" "Parm1:1998" "Parm3:Ido Millet"

Notes:

- Visual CUT handles the necessary data type conversion to match the parameter data type.
- All command line arguments must be **enclosed in double quotes** and **separated by a single space**.

Range and Multi-Value Parameters

Visual CUT supports all parameter types including multi-value, range, and mixed parameters. To learn how such parameter values can be specified via command line arguments, save the report settings in Visual CUT, open **Visual CUT.mdb** in MS Access and observe the parameter values saved in the **Report_Opt** table.

A multi-value discrete parameter value is specified as follows:

... **"Parm1:Competition:::Gloves:::Helmets"**

A range parameter (in this case a date range) is specified as follows:

... **"Parm1:7/15/1996>>>7/15/2003>>>3"**

The **3** at the end indicates the start and end points are included.
A **0** at the end would indicate the start and end points are NOT included.
A **2** at the end would indicate the start point is included and the end point is not.
A **1** at the end would indicate the start point is not included and the end point is.

Add 4 to these values if there is no Upper Bound.
Add 8 to these values if there is no Lower Bound.

For example, this would indicate all dates up to, and including 7/15/2003:

... **"Parm1:12:00:00 AM>>>7/15/2003>>>9"**

The 12:00:00 AM value is just a place-holder. Any date value would work (will be ignored).

Request User Input for Certain Parameters

You can use "ParmN:**[?]**" command line arguments to indicate that Visual CUT should prompt the user for certain parameter values. For example,

---

... "Parm1:Today" **"Parm3:[?]"**

Would set the first parameter to today's date, prompt the user for the 3rd-parameter, and use saved parameter values for all other parameters. This is useful when Visual CUT is called from a command line and the user needs to interactively override saved parameter values.

Null Values

Null parameter values (for Stored Procedures) are specified in command lines by using the constant **[VC_NULL]**. For example, to specify that the first parameter value is null, use: "Parm1:**[VC_NULL]**"

Date Constants

When scheduling reports that have Date or DateTime parameters, you can set the parameter to dates relative to the current date. Visual CUT can do this for **discrete** or **range** date parameters. The supported constants are:

1. TODAY -or- YESTERDAY
2. TODAY_PLUS_**N** -or- TODAY_MINUS_**N** -or-
   TODAY_MINUS_**N**_MINUS_**M** -or-
   TODAY_MINUS_**N**_MINUS_**M_EOM** -or-
   TODAY_MINUS_**N**_MINUS_**M_SOM**

3. START_MONTH_PLUS_**M** -or- START_MONTH_MINUS_**M**
4. END_MONTH_PLUS_**M** -or- END_MONTH_MINUS_**M**

5. START_YEAR_PLUS_**Y** -or- START_YEAR_MINUS_**Y**
6. END_YEAR_PLUS_**Y** -or- END_YEAR_MINUS_**Y**
7. Nth_**N**_PLUS_**M** –or- Nth_**N**_MINUS_**M**
8. LAST_MM_DD –or- NEXT_MM_DD

9. Now_Plus_**S** -or- Now_Minus_**S** -or- Now_GMT_Plus_S -or- Now_GMT_Minus_S

Where **N=days, M=Months**, **Y=Years**, **and S=seconds** to be added or subtracted.
For example, if the current date is **March 6, 2004** then:

Today = 3/6/04
Nth_16_MINUS_1 = 2/16/04 (the 16th of the previous month)
Today_Minus_3 = 3/3/04
Last_04_01 = April 1, 2003
[?
End_Month_Minus_1 = 2/29/04
Start_Year_Plus_0 = 1/1/04
Start_Year_Minus_1>>>Today>>>3 = Inclusive **range** of [**1/1/03** to **3/6/04**]

In the case of Today_Minus_**N**_Minus_**M**, **N** is the Days and **M** is the Months, so:
Today_Minus_**1**_Minus_**2** = 1/5/2004 (one day and two months earlier)
Adding _EOM or _SOM to the end of a Today_Minus_**N**_Minus_**M** constant returns the End-of-Month or Start-of-Month, so:
Today_Minus_**1**_Minus_**2_SOM** = 1/1/2004 (Start of Month for 1/5/2004)
Today_Minus_**1**_Minus_**2_EOM** = 1/31/2004 (End of Month for 1/5/2004)

For DateTime parameters, **Now_Plus_S** or **Now_Minus_S**, returns the current datetime adjusted by the number of specified seconds. For Time parameters, this argument returns just the current time adjusted by the number of specified seconds. For example, if the current datetime is **June 17, 2008, 5:22:38 PM** then **Now_Minus_3600** returns **June 17, 2008, 4:22:38 PM** for a **DateTime** parameter and **4:22:38 PM** for a **Time** parameter.

One way to use a date constant is to specify it as the parameter value in a command line invocation of Visual CUT. For example:

"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Test\Report.rpt" "Parm1:Today"

Another option is to open **Visual CUT.mdb** and entering it directly in the appropriate parameter column within the **Report_Opt** table. Note that such a manual entry in the **Report_Opt** table would be overwritten if you interactively open the report in Visual CUT and click SAVE.

Benefits: these date constants allow you to use the same report interactively (specifying any date as the parameter value) as well as in scheduled mode. This can also lead to faster report execution since using date functions in the record selection formula within the report can force record selection to be performed by Crystal instead of by the DBMS.

Note: for DateTime parameter, the Time portion of the parameter value is set by these constants to 12:00:00 AM (start of the day). **To set a DateTime parameter to the end of the day (11:59:59 PM)**, you must include the parameter name in a DataLink_Viewer.ini entry under the [Options] section like this (the parameter names are separated by ||):

```
--------------
[Options]
Date_Constants_EndofDay_Parameters={?To_Date}||{?EndDate}
-------------
```

Custom Calendars

If you need to set date parameters to the start or end date of custom periods (for example, Fiscal Weeks or Quarters) relative to a current or shifted date, you specify the start dates of the periods in the DataLink_Viewer.ini as follows:

```
[Custom_Calendars]
FiscalQ=1/1/2013>>4/1/2013>>7/1/2013>>10/1/2013>>1/1/2014
```

In the example above the name of the custom calendar is FiscalQ and you can specify parameter values like this: "Parm1:**FiscalQ_Start_RelativeTo_Yesterday**"
FiscalQ is the name of the custom calendar, **Start** requests the start date of the period within which the relative date falls. If you specify **End**, the date just before the start on the next period is returned. **RelativeTo** is a fixed word. The **date constant** can be any date constants as described in the Date Constants section above. Assuming today is 6/7/2013:
"Parm1:**FiscalQ_End_RelativeTo_Today_Minus_5**" **would return** 6/30/2013
"Parm1:**FiscalQ_End_RelativeTo_Start_Year_Minus_1**" **would return** 3/31/2012

Note: as demonstrated by the last example above, **if the relative date falls outside the boundaries of the calendar, the custom calendar years are shifted until the relative date is within the calendar**. This **allows you to set the custom calendar and not worry about updating it again** (if the dates remain the same across years).

Adjusting Data Constants for Day of Week

Visual CUT allows you to set a date parameter to the first day of week (DOW) before or after a specified date constant. For example, the first Monday in the previous month.

The syntax options are as follows:

```
DayOfWeek[>]Date_Constant      for first target DOW after the date constant.
DayOfWeek[<]Date_Constant      for first target DOW before the date constant.
```

Or, same as above except that date returned from date constant is used if it falls on target DOW:

```
DayOfWeek[>=]Date_Constant
DayOfWeek[<=]Date_Constant
```

Examples, assuming today is Tuesday, March 10, 2015

```
"Parm1:Wednesday[>]Today"        => March 11, 2015
"Parm1:Wednesday[<]Today"        => March 04, 2015
"Parm1:Tuesday[>]Yesterday"      => March 10, 2015
"Parm1:Tuesday[>=]Today"         => March 10, 2015
```

Number Constants

When scheduling reports that have a Number parameter, you may want to set the parameter to a constant reflecting a year or a month relative to the current year or the current month.
Visual CUT can do this for **discrete** or **range number** (not currency) parameters.

The supported constants are:

1. MONTH_PLUS_**N** -or- MONTH_MINUS_**N**
2. YEAR_PLUS_**N** -or- YEAR_MINUS_**N**
3. YEAR_AT_PLUS_MONTHS_**N** -or- YEAR_AT_MINUS_MONTHS_**N**
4. YearMonth_AT_PLUS_MONTHS_N –or- YearMonth_AT_MINUS_MONTHS_N

Where N is the number of months/years to be added/subtracted from the current month/year.

For example, if the current date is **January 6, 2005** then:

Month_Plus_0 = **1**
Month_Minus_1 = **12**
Year_Plus_0 = **2005**
Year_Minus_2 = **2003**
Year_AT_Minus_Months_1 = **2004**
YearMonth_AT_MINUS_MONTHS_3 = **200410** (year=**2004** and month = **10**)

One way to use a Number constant is to specify it as the parameter value in a command line invocation of Visual CUT. For example:

"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Test\Report.rpt" "Parm1:Year_Plus_0"

Another way of doing this is by opening **Visual CUT.mdb** and entering it directly in the appropriate parameter column within the **Report_Opt** table. Note that such a manual entry in the **Report_Opt** table would be overwritten if you interactively open the report in Visual CUT and click SAVE.

Benefits: these constants allow you to use the same report interactively as well as in scheduled mode. This can also lead to faster report execution since using formulas within the report can force record selection to be performed by Crystal instead of by the DBMS.

Reports with More Than 8 Parameters (restriction removed, 6.2002)

This restriction is removed starting March 19, 2010. Any parameters beyond the first 7, including unlinked subreport parameters, are now saved inside the Parm8 column (in the Report_Opt table) for later handling by Visual CUT. You may still specify main report parameters via command line arguments such as
…."Parm9:Some Values" "Parm10:Some Values"

Using Parm8 to Specify all Extra Parameter Values

Let's refer to all independent (unlinked) subreport parameters as well as all parameters beyond the first 7 main report parameters as "Extra" parameters. As mentioned above, starting with version 6.2002, all extra parameter values are saved inside the Parm8 column inside the Report_Opt table in Visual CUT.mdb. The syntax for holding these values looks like this:

```
[Parm_8]:Suppress
-----
[Parm_9]:2010
-----
[Parm_10]:4/1/2010>>>4/30/2010>>>3
-----
[Parm_11]:Competition:::Gloves:::Helmets
-----
[Parm_1_{Subreport1Name.rpt}]:2010
-----
[Parm_2_{SomeSubreport1Name.rpt}]:101
```

As you can see, main report parameters are identified by their number and independent (unlinked) subreport parameters are also specified by the subreport name. The parameters are separated by "-----" delimiters.

If you wish to use command line arguments to override the stored Parm8 values, you can use the Parm8 command line argument using the same syntax. For example (all in 1 line of course):

```
"Parm8:[Parm_8]:Suppress-----[Parm_9]:2010-----[Parm_10]:START_MONTH_MINUS_1
>>>END_MONTH_MINUS_1>>>3-----[Parm_11]:Competition:::Gloves-----
[Parm_1_{Subreport1Name.rpt}]:2010-----[Parm_2_{SomeSubreport2Name.rpt}]:101"
```

Using PowerShell to Set Relative Date Parameters and Call Visual CUT

The following PowerShell code sample was contributed by Roger Dearnaley, IT Manager at Sataria. It demonstrates how PowerShell calls Visual CUT (using the **&** symbol to trigger the command line) after applying relative date logic to set parameter values.

Similar logic can be achieved using regular batch files, but this sample may help other users who are interested in moving from batch files to PowerShell files.

```
<#
emp perf reports.ps1  Roger Dearnaley, Sataria, November 2013
this script will call two Crystal Reports using Visual CUT
It will pass the previous weekday (i.e. Saturday=Friday, Sunday=Friday, Monday=Friday,
Tuesday=Monday.....Friday=Thursday)
the daily report will run on Monday-Friday and weekly will only run on Monday
#>

[DateTime]$now = ([DateTime]::Now)
[DateTime]$prevWorkDay = $now
switch (($now.DayOfWeek))
   {
      "Sunday"
          { $prevWorkDay = $now.AddDays(-2) }
      "Monday"
          { $prevWorkDay = $now.AddDays(-3) }
      default
          { $prevWorkDay = $now.AddDays(-1) }
   }

$exe = "C:\Program Files (x86)\Visual CUT 11\Visual CUT.exe"

#run daily report
[Array]$params = '-e', '"\\Mufasa\Publicdocs\WMS Reports\Operations\daily emp perf.rpt"',
[string]::Format('"Parm1:{0:MM/dd/yyyy}"',$prevWorkDay)
& $exe $params;

#run weekly report on Monday only
if ($now.DayOfWeek -eq "Monday")
{
   [Array]$params = '-e', '"\\Mufasa\Publicdocs\WMS Reports\Operations\weekly emp
perf.rpt"', [string]::Format('"Parm1:{0:MM/dd/yyyy}"',$prevWorkDay)
   & $exe $params;
}
```

Using Command Line Argument to Set Formula Expressions

You can set formula expressions via a command line argument provided that:

- The formula name(s) must begin with a ^ character.
- Any double-quotes are specified as [dblq]
- Only main report formulas are targeted

The command line argument starts with **Set_Formulas1:** followed by pairs of **Formula Names** and **Formula Expressions**. The name is separated from the expression by >>>. Each pair is separated from the following one by ||| like this:

… **"Set_Formulas1:Name1>>>Expression1|||Name2>>>Expression2"**

For example, the following command line argument would set the first specified formula expression to the string "Ido" and the second formula to the numeric expression of 2 + 2:

… **"Set_Formulas1:{@^MyName}>>>[dblq]Ido[dblq]|||{@^TwoAndTwo}>>>2+2"**

Note: this functionality is not available in Visual CUT 8.5

Arguments to Specify Printer Destination

If you turn on the "Print (if scheduled)" option, Visual CUT sends the report to the printer associated with the report. You can override the printer destination by specifying a different printer name using a command line argument:

"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt" "Parm1:1998" "Printer:HPLaser1"

Note that the syntax is constructed as the word "Printer:" followed by the printer name.

If you use "**Default**" as the printer name, the report gets printed to its default destination.

If instead of "**Printer:**…" you use "**Printer_Only:**….", Visual CUT limits the processing to printing only. Exporting, bursting, and e-mailing options saved for the report in Visual CUT are ignored. This allows you to invoke Visual CUT processing for printing only via one command line and invoke full processing via another command line.

When a Command Line includes "**Printer:**" or "**Printer_Only:**" argument, Visual CUT always sends the report to the printer even if the "Print (if scheduled)" option is turned off for this report.

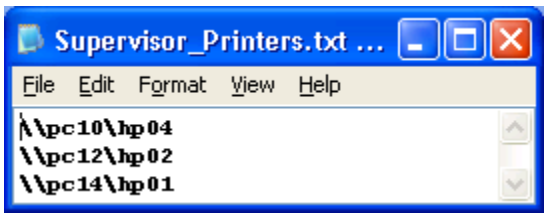
Printing to Multiple Printers

If you need to schedule or invoke printing of the same report to multiple printers on your network, you can specify an unlimited number of "**Printer:**" and/or "**Printer_Only:**" command line arguments and Visual CUT would send the report to all these printers. This improves performance compared to specifying each printer destination in a different command line because the report gets retrieved and processed only once.

Using a Text File to Specify Multiple Printers

If you have a text file containing a list of printer destinations, such as shown below, you can instruct Visual CUT to print the report to all these printers using the key word "**List_File:**" followed by the path & name of the text file.
"Printer:**List_File:c:\temp\Supervisor_Printers.txt**"

As usual, you can dynamically control the list of printers to be used by referencing field or formula names from the Crystal report. For example:
"Printer_Only:**List_File:{@Printers_File}**"



Arguments to Specify Printer Bursting Destination

If you need each group level 1 of your report to be printed separately (e.g., to a fax printer driver or for automatic stapling), you must have the Export and Burst options turned on in the Visual CUT saved settings for that report. Then, use the following command line argument structure:
"C:\Program Files (x86)\Visual CUT\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt" "Parm1:1998" "Printer_Burst:\\pc10\hp04"

Note that the syntax is constructed as the word "**Printer_Burst:**" followed by the printer name. If instead of "**Printer_Burst:**…" you use "**Printer_Burst_Only:**….", Visual CUT will skip the export bursting processing and would just burst the report to the printer.

If you use "**Default**" as the printer name, the report gets printed to its default destination.

The **printer name argument is dynamic** (field/formula names are replaced with values from the report). This means you can create a formula that returns a different printer name for different group values, place that formula (suppressed) in GH1 or GF1, embed that formula as the printer name argument, and sent the printouts for different groups to different printers. For example:
"C:\Program Files (x86)\Visual CUT 11\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt" "Parm1:1998" "Printer_Burst_Only:{@Group_Printer_Name}"

Specifying Number of Copies

To control the number of printed copies, you can use a "**Print_Copies**" argument. The syntax is constructed as the word "**Print_Copies**", followed by a colon and the number of desired copies. For example: **"Print_Copies:3"**

Setting Custom Text for each Print Copy

When printing directly to a printer (not via PDF_Print), each print copy can have custom text (such as 'Copy 1 of 2', 'Copy 2 of 2'). Simply place a formula called **{@Print_Copy_Template}** on the report canvas. For each printed copy, Visual CUT would first update the text in that formula by replacing **[[N]]** with the copy number and **[[M]]** with the total number of copies.
See this demo image.
Note: you can dynamically control the number of copies by referring to a Crystal formula/field in a "Print_Copies" argument (e.g. "Print_Copies:{OrderDetail.Quantity}" or by simply dragging & dropping the field/formula into the Copies option (see image).

Arguments to Specify User ID & Password

In some rare cases you may want to override the (encrypted) User_ID & Password information that Visual CUT stores for each report. You can do this by specifying Command Line arguments:

"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Test\Report.rpt" "user_id:dba" "password:sql"

Setting Encrypted Password Entries

The **Encrypted_Password_Set_Entry** command line argument can automate encrypted storage (within DataLink_Viewer.ini) of passwords. This allows an administrator to set or change encrypted passwords for DataLink Viewer or Visual CUT that can later be referenced from command line arguments.

To generate such entries in a targeted ini file, you call Visual CUT using a command line (all in 1 line) argument as follows:

"C:\Program Files (x86)\Visual CUT 11\Visual CUT.exe" "**Encrypted_Password_Set_Entry: H:\DataLink_Viewer.ini>>Options>>Encrypted_Password_FTP>>sesame**"

After the **Encrypted_Password_Set_Entry** key word and the colon, the 4 arguments are separated by ">>" as a delimiter:

1. The **path & name of the ini file**
2. The **ini section name** (typically '**Options**')
3. **The Password Name**
4. **The Password to be Encrypted**

Database Choice Functionality (Command Line / GUI)

In some cases, you may want to use the same report to connect to different data sources, such as a **testing** or **production** server. While each report stores connection properties for only one default data source, Visual CUT allows you to use Command Line arguments to specify a different data source.

The command line argument structure is as follows:

Selecting an Alternative ODBC Data Source

… "ODBC_DSN:Data Source Name"
or
… "ODBC_DSN_From_To:Old_DSN1>>New_DSN1||Old_DSN2>>New_DSN2"

The **ODBC_DSN** argument overrides all ODBC DSNs used in the report by the new DSN
The **ODBC_DSN_From_To** argument overrides only tables that use the old DSN.

Notes:

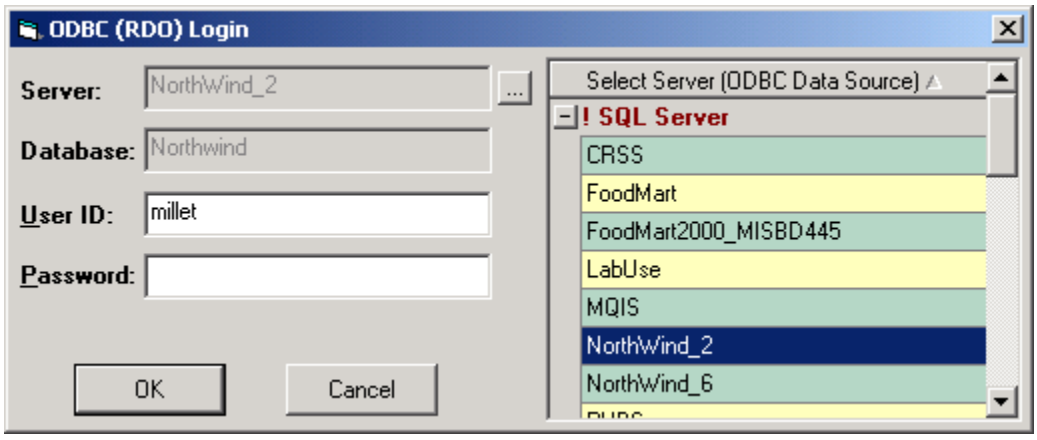
1. You can combine this functionality with the **user_id** & **password** command line arguments described in a previous section.

2. The target **Data Source Name** must exist on the PC running Visual CUT. If it doesn't, Visual CUT reverts back to running the report against the ODBC DSN the report was designed for (a warning message is added to the failure.log file).

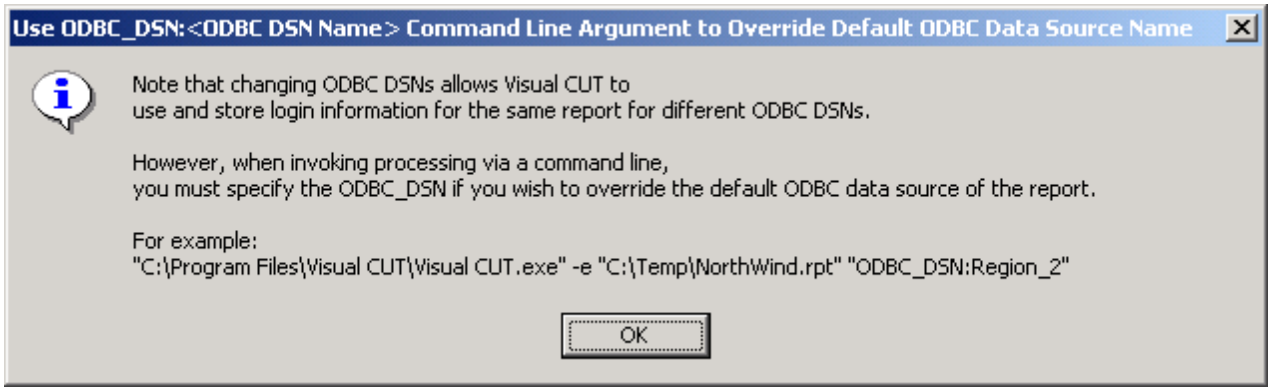
3. **ODBC_DSN_From_To** support **multiple pairs** separated by "||" as shown in the example above. This addresses scenarios where a report uses multiple ODBC DSNs (e.g. when subreports use different DSNs).

4. The **ODBC_DSN_From_To** is not available in the Crystal 8.5 version of Visual CUT.

5. If you don't specify User_ID & Password command line arguments, and the data source requires a user id & password, you should let Visual CUT "load" the authentication information by running the report against the alternative ODBC data source. You can do that by clicking the button to the right of the Server name in the login dialog. This expands the display to include a listing of all available ODBC data sources (grouped by ODBC driver type). The original ODBC data source is initially selected and the driver group it belongs to is expanded (and prefixed with a "!").



6. If you select another ODBC data source, Visual CUT explains the functionality:


Use ODBC_DSN:<ODBC DSN Name> Command Line Argument to Override Default ODBC Data Source Name
Note that changing ODBC DSNs allows Visual CUT to
use and store login information for the same report for different ODBC DSNs.
However, when invoking processing via a command line,
you must specify the ODBC_DSN if you wish to override the default ODBC data source of the report.
For example:
"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Temp\NorthWind.rpt" "ODBC_DSN:Region_2"
OK

Overriding the Database Specified in the Report or ODBC DSN

For ODBC data sources, you can enter a database name into the login dialog if you wish to override the database specified in the ODBC data source or in the report itself. This is useful for situations where the same database (e.g., MS SQL Server) contains multiple databases each with the same table structure. If the number of such databases is large, creating a dedicated ODBC DSN for each and using the select ODBC DSN functionality may be too tedious. Instead, you can directly type in the database name in the login dialog.

To enable database name input into the database field in the login dialog, you need to set the **Override_ODBC_DSN_Database** entry under the [Options] section in the DataLink_Viewer.ini file to TRUE, like this:

[Options]
Override_ODBC_DSN_Database=TRUE

Note that if that option is set to TRUE, the database specified in the ODBC DSN can no longer override the database specified in the report (if the user doesn't type in a Database, the specified Database remains the one used in the report.

Note: This functionality is not available in the Crystal 8.5 version of Visual CUT.

Overriding ODBC Table Location (.CSV Files as Data Source)

This functionality was added for a user who needed to call DataLink Viewer from his application via a command line. The data source was **.csv** files via ODBC and a command line argument was needed to control what **.csv** file should be used (.csv files gets generated and named uniquely by another application).
For example, a report designed to use **Risk.csv**, the following command line argument:
… **"Table_From_To:Risk.csv>>Risk2.csv||old.csv>>new.csv"**
would cause Visual CUT to retrieve the data from **Risk2.csv** instead.

Within each pair of From/To directives, the 'From' location is separated by a '>>' from the 'To' location. The pairs are separated by a "||" from each other.

Notes:

1. If any of the ***From*** table name is not found in the report, the process generates a failure message indicating what ***From*** table name were unmatched with report tables.

2. New Table Name should not contain dots (before the .csv). Underscores are fine.

3. This functionality is not available in Visual CUT 8.5

Overriding the Server in Native Oracle Connection

When a report uses a **native connection** to Oracle, you can **edit the Server name** in the login dialog and run the report against a different server (rather than the one the report was designed against).

Alternatively, if you are launching a report from a command line, you can override the Oracle server name by using the "**Oracle_Server:**" command line argument. For example:

… -v "C:\temp\test.rpt" "**user_id:dba**" "**password:sql**" "**Oracle_Server:Server2**"

Note: This functionality is not available in the Crystal 8.5 version of Visual CUT.

Selecting an Alternative SQL Server – OLE DB Data Source

… "Connect_To_SQLOLEDB:DataSource>>InitialCatalog>>Integrated_Auth"

The parameters (after the ":") are separated by a ">>" and are as follows:
1. **DataSource** is the **Server Name** the report should connect to.
2. **InitialCatalog** is the **Database Name** within the given server.
3. **Integrated_Auth** is a **True** or **False** argument indicating if Integrated Authentication should be used (if TRUE, user_id & password are ignored).

Note: This functionality is not available in the Crystal 8.5 version of Visual CUT.

Changing Folder Location for Access/Excel/Pervasive/ACT! Files

If your report uses the native connection to MS Access, Excel, or Pervasive (ddf), you can control the location of the database files using the following section in DataLink_Viewer.ini

```
-----------------------------------------------------------------
[Database_Path_Selection]
Paths = C:\Old\xtreme.mdb>>C:\New\xtreme.mdb||C:\a\test.mdb>>?
// Or, in the case of Pervasive ddf files:
// for ACT! .pad files: Paths = E:\DB1\DB1.pad>>E:\DB2\DB2.pad
// Paths = E:\Jobtrack\FILE.DDF>>?
-----------------------------------------------------------------
```

Alternatively, you can specify the desired change using a command line argument such as …**"Database_Path:C:\Prop1\FILE.DDF>>c:\Prop2\File.DDF"**

As demonstrated above, the Paths entry may contain multiple pairs of old>>new paths. Each pair specifies the **old path** followed by a "**>>**" separator, followed by the new path. The pairs are separated by "||"

If the new path is blank, or if it contains just a question mark, Visual CUT will prompt the user to select a new path. If the directive is from the ini file (rather than from a command line argument), the user choice will be recorded in the ini file and the new path will be used without prompting the user in future runs of reports that use the source path. If the new path has one question mark (**?**) followed by a valid path, that path will be the default location in the dialog asking the user to select a path. If instead of a single question mark, the new path starts with a double question mark (**??**), DataLink Viewer will always prompt the user to select a path each time a report using the old path runs. The user choice will be added after the **??** and will become the new default value in the path selection dialog.

If the report uses a database file that can't be found on the machine, and no command line argument or ini entry was provided, Visual CUT prompts the user to select a valid location, creates the **[Database_Path_Selection]** section (if it doesn't already exist) and creates/adds the pair information to the **Paths** entry.

Note: this functionality is not available in the Crystal 8.5 version of Visual CUT.

Arguments to Specify Export Format

If you need to export a report to format and file name that are different from those saved for the report within Visual CUT, you can use the **Export_Format** in combination with the **Export_File** command line argument. For example:

"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt" "Export_Format:Excel 97" "Export_File:c:\temp\test.xls"

The syntax is constructed as the word "**Export_Format:**" followed by the export format name (see valid export format options in the Visual CUT drop-down list for export formats).

Dynamic Export Format: you can use field/formula names within the Export_Format command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT). The dynamic content of these fields/formulas would be substituted into the command line argument. For example, you could use the following command line:

"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt" "Export_Format:{@Format}"

The {@Format} formula may return different export formats for different groups, so using a parameter (that influences the {@Format} formula) or during bursting, you can dynamically control what export format is used. For example, you may burst invoices to some customers as PDF and to other customers as Excel files.
In such situations, you will need to construct the export file name to include a formula for controlling the file extension to match the export format. For example:

"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt" "Export_Format:{@Format}" "Export_File:c:\temp\Invoice_for_{CustName}.{@ext}"

Argument to Specify Email Priority

Email priority can be specified via a command line argument using the following syntax:
… "Email_Priority:Priority_Constant"
The syntax is constructed as the word "**Email_Priority:**" followed by one of the following priority constants**: Highest, High, Normal, Low**, or **Lowest**

As usual, you can dynamically control the email priority by creating a formula within the report that would determine the email priority (for example, if the information shows an extreme exception, increase the email priority to 'Highest'). Then, refer to the formula in the command line argument and Visual CUT would substitute the value of that formula. For example:
… "Email_Priority:{@Message_Priority}"

Argument to Specify Email Headers

Custom Email headers can be specified via a command line argument using the following syntax:
… "Email_Header:header1:::header2:::header3"
The syntax is constructed as the word "**Email_Header:**" followed by one or more custom email headers. If you are specifying more than one header, they must be separated by ":::".
For example:
… "Email_Header:Sensitivity: Private"
… "Email_Header:Sensitivity: Private:::X-Sensitivity: 3"

As usual, you can dynamically control the content of the custom email headers by referencing field or formula names from the Crystal report. For example:
… "Email_Header:Sensitivity: {@email_Sensitivity}"
or, for return-receipt request, which asks the receiving email server to confirm receipt:
…"Email_Header:Return-Receipt-To:{@CRM_Email}"
or, to ask for confirmation the email has been opened by the recipient:
…"Email_Header:Disposition-Notification-To:{@CRM_Email}"

Arguments to Specify Email Options

If you need to override the values specified in the emailing options in the 3rd tab of Visual CUT you can use the following command line arguments:

1. **Export_File**
2. **Export_Mode** (**"Burst"** or **"Whole" or blank**)
3. **Email_To**
4. **Email_From**
5. **Email_Reply_To**
6. **Email_CC**
7. **Email_BCC**
8. **Email_Attach**
9. **Email_Subject**
10. **Email_Message**
11. **Email_Mode ("Burst" or "Whole" or blank)**
12. **Email_SMTP_Server**

For example, to override the export file and email_to option, use a command line such as:
"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt" "Parm1:1998" "Export_File:c:\temp\test.pdf" "Email_To:ixm7@psu.edu"

To override a saved value with a blank, use **[VC_Blank]** as in: … "Email_CC:**[VC_Blank]**"

Arguments to Process Reports with No Settings

Imagine you have an application that needs to trigger Visual CUT processing for reports that have no saved settings within Visual CUT. Visual CUT can handle such cases provided all the necessary options are specified via command line arguments.

For example, the following command line (all in one line) would trigger exporting and emailing of Sample.rpt. The highlighted arguments (Export_Mode and Email_Mode) control what exporting and/or emailing process is required. They accept values of "**Burst**" or "**Whole**" and if one of them is omitted, then that aspect of processing will simply not take place.

"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Test\Sample.rpt" "**Parm1**:1997" "**Export_Format**:Adobe Acrobat (pdf)" "**Export_File**:c:\temp\{Product_Type.Product Type Name}.pdf" "**Export_Mode**:Burst" "**Email_Mode**:Burst" "**Email_From**:ixm7@psu.edu" "**Email_To**:ixm7@psu.edu" "**Email_Subject**:Sales Information for {Product_Type.Product Type Name}" "**Email_Attach**:c:\temp\{Product_Type.Product Type Name}.pdf" "**Email_SMTP_Server**:127.0.0.1"

Calling Visual CUT From Another Application

Using command lines allows other applications to trigger Visual CUT processing.

Here's a code example of calling Visual CUT from a **Visual Basic** application and specifying a parameter value. Note that double quotes are "escaped" by using `""` instead of `"`.

```
Dim ls_temp As String
ls_temp = "c:\Program Files\Visual CUT 9\Visual CUT.exe " & _
"-e ""c:\temp\Sales.rpt"" ""Parm1:2005"""

RetVal = Shell(ls_temp)
```

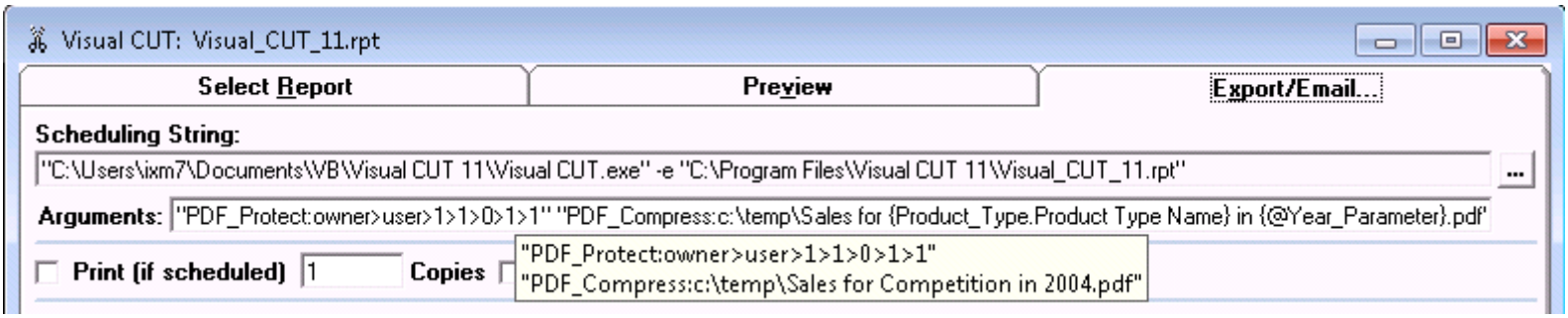
Here's another code example for calling Visual CUT from a **vba** event and dynamically setting the report path, name, and parameter value:

```
Private Sub Combo0_AfterUpdate()
Dim rs As Object
Set rs = Me.Recordset.Clone
rs.FindFirst "[txtReportName] = '" & Me![Combo0] & "'"
If Not rs.EOF Then Me.Bookmark = rs.Bookmark

Dim myreport As String
Dim stAppName As String
Dim myvalret As String
' me.fullrep is a field that concatenates the report path & name
myreport = Me.fullrep
myvalret = Str(MyCaseno)
stAppName = "C:\Program Files\Visual CUT 9\Visual CUT.exe" & _
           " -e """ & myreport & """ ""Parm1:" & myvalret & """"
DoCmd.Close
Call Shell(stAppName, 1)
End Sub
```

Note: your application can **monitor success/failure** of the command line call using the functionality described in the "***Job Status Functionality***" section of this user manual.

Specifying Arguments from the GUI

Starting with version 6.3002 you can save and apply command line arguments through the GUI. The Arguments field shown below gets saved into the "Arguments" (memo) column in the Report_Opt table in Visual CUT.mdb.



A minor benefit is that you can **simplify command lines in batch files** by using the minimal command line and storing all the arguments in the database.

A major benefit is that **you can take advantage of these arguments even during interactive use**. For example:

- Use **Skip_Recent** to avoid duplicate emails during interactive processing
- Use any of the PDF, XLS, Word, … options such as password-protecting files during interactive processing
- Request printing (e.g., `"Printer_Burst:Default"`) while also Exporting and Emailing
- Etc.

Notes:

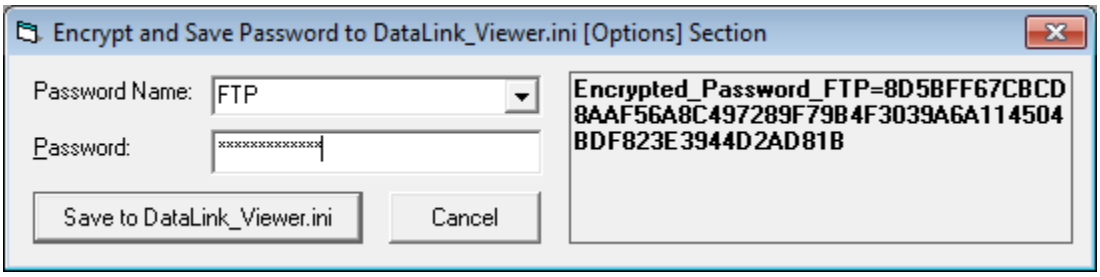
1. A double-click on the Arguments field opens the text in a multi-line editing mode. This makes it easier to view and change the arguments. When clicking OK, the content gets saved to the field as a single line.
2. Command line processing honors these saved arguments but can override them.
3. Users of prior versions who wish to take advantage of this new feature should add an **Arguments** column (**Memo** data type**)** to the **Report_Opt** table in **Visual CUT.mdb**

Referring to Saved Encrypted Passwords

This section describes how to centralize and protect passwords by avoiding specifying them directly inside command line argument. Instead, you can name, encrypt and store the passwords inside DataLink_Viewer.ini. Besides protecting your passwords, this also allows you to change the passwords in one location, instead of in multiple command line arguments.

Visual CUT already encrypts passwords for connecting reports to data sources, sending emails, exporting to ODBC, and capturing incoming emails. But plain password might be used in the following command line argument: After_Success_SQL, Email_Password, FTP_Upload, SFTP_Upload, Password, PDF_Protect, PDF_Merge, Word_Protect, XLS_Protect, XLS_Protect_Sheets, XLS_Modify_Protect, ZIP_Files. In addition, Email_To, Email_CC, and Email_BCC can contain passwords when using the option to get email addresses via an ODBC call.

To protect and centralize password in the later scenarios, go to the Process tab in the Options dialog and click on the '***Encrypt & Save Password***' button. This provides the following dialog:



In the case above, I named the password for my FTP_Upload as "**FTP**". The ini file entry is always named as "**Encrypted_Password_**" followed by the password name you specify. So in my case, the full name is **Encrypted_Password_FTP**. From that point on, I can refer to that password inside my command line arguments as ***Encrypted_Password_FTP*** and Visual CUT would make the appropriate substitution. For example:
"**SFTP_Upload**:22>>{@server}>>PW>>ido>>**Encrypted_Password_FTP**>>>>>>c:\temp*.html>>>>10"

If you need to change a previously saved password, use the drop-down to select the previously saved password name and enter a new password.

Other Options and Features

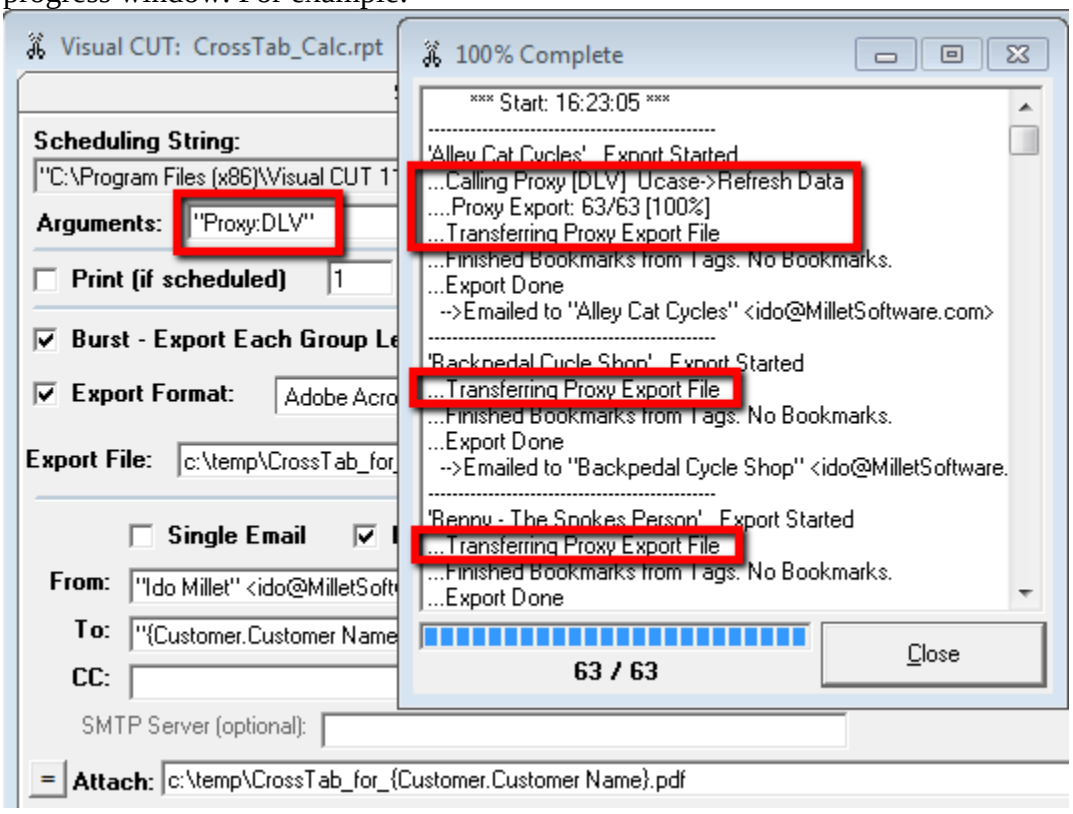
Delegating Exporting/Bursting to DataLink Viewer 2011

Visual CUT 11 can delegate exporting/bursting operations to DataLink Viewer 2011 in order to support Crystal 2008/2011/2013 features:
1. Calculated CrossTab members
2. Dissociate Formatting Page Size and Printer Paper Size
3. Auto-Arrange option for chart layouts
4. The group expert options of 'New Page After N Visible Groups'
5. Mixed portrait & landscape sections
6. Embedded Flash objects.

The command line argument to request this is "Proxy:DLV". If the report has saved data and you wish to not refresh that data, specify dlv in lower case, like this: "Proxy:dlv"

All the parameters and login info used for running the report in Visual CUT are automatically applied to the DataLink Viewer 2011 process as well.

During bursting, the DataLink Viewer 2011 process communicates back to the Visual CUT progress window. For example:



Triggering a Batch File with Dynamic Content Before/After Export

Imagine that you need to export some reports and then trigger some command line processing such as packaging the files into a .RAR archive before uploading the archive to an FTP server.

The **After_Export_Batch** command line argument allows you to insert batch file operations **after each successful whole report export. Note: After_Burst_Batch** does the same after each **group** export step). Visual CUT then waits for the batch file to complete processing before continuing with any other Visual CUT processing such as emailing or uploading to FTP.

The **Before_Export_Batch** command line argument does the same before Visual CUT exports the report. A typical use is to archive previously exported files.

The key feature is that **you can embed field/formula names within the batch file** just as you can within the Visual CUT 3rd tab options. **Visual CUT substitutes the appropriate values for these field/formula names before launching the batch file processing**. A typical use for this is to make sure dynamic export file names are used within the batch file.

For example, using the following command line (all in one line):

```
"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\test.rpt"
"After_Export_Batch:C:\test.cmd"
"FTP_Upload:Passive_1>>ftp://server1.RSB.com>>user1>>Pass1>>
c:\test{[yy]}{[MM]}{[dd]}.rar>>MyFolder"
```

---

would cause Visual CUT to:

1. Export the test.rpt to a desired format and dynamic file name

2. Look into the c:\test.cmd batch file to **locate and temporarily replace Crystal fields/formula references with their dynamic values**.
   For example, the following batch file:

   ```
   "C:\Program Files\WinRAR\rar.exe" a "c:\test{[yy]}{[MM]}{[dd]}.rar"
   "c:\temp\*{[yy]}{[MM]}{[dd]}.txt"
   ```

   Would be executed as if it was

   ```
   "C:\Program Files\WinRAR\rar.exe" a "c:\test101014.rar"
   "c:\temp\*101014.txt"
   ```

3. Visual CUT would then continue by executing the FTP upload the resulting RAR file.

Note: for scheduled processing under Windows 2008 you may need to set the batch file properties to: '***Run this program in compatibility mode for***' Windows XP (Service Pack 3). Use the option (a button in the Properties dialog) to set the compatibility **for all users**.

Triggering Reports Based on Database, File, and Email Events

The After_Export_Batch argument allows one report to monitor a database, file, or email (Crystal can use the file system or Exchange folders as data source), and locate new entries based on record selection logic or based on Visual CUT Skip_Recent (or process log) logic. Then, if a new and valid Database/File/Email event is detected, export to a dummy file and trigger a second report, passing to it (via a batch file) parameters from the first report.

Printing

Bursting to Printer

You may want to burst a report directly to a printer. For example, you may want each group to become a separate print job so it gets stapled.

There are 3 options for achieving this.

Method 1
Use **Printer_Burst** or **Printer_Burst_Only** command line arguments. For detail, see:
Using Command Line Arguments to Specify Printer Bursting Destination

Method 2
Burst to PDF files and use **PDF_Print** command line arguments to print them. For detail, see:
Printing PDF Files

Method 3
Export to **Printer (Default)** or **Printer (Specified)** as the export format.
This allows print bursting to be initiated/stopped interactively. If the **Rename_Printer_Jobs** option in the DataLink_Viewer.ini file is set to True, the printer queue shows the group value for each burst cycle print job.

When using **Printer (Specified)** as the export format, the export file name option controls the name of the printer. Since this option can be dynamic (contain field/formula reference), this means that each group can be printed to a different printer.

Users who don't see these export format options in the drop-down should add two new rows to the **Export_Opt** table in the **Visual CUT.mdb** Access database:

Export Constant	Export Name
Printer_Default	Printer (Default)
Printer_Specified	Printer (Specified)

Interweaving Burst Printouts From Multiple Reports

Imagine that you need to burst a report such that each group results in a check printed on special paper using Landscape orientation followed by supporting documentation using Portrait orientation on another type of paper. Or perhaps you need to burst 3 different reports (sales, commissions, and product returns) to sales representatives and you want the output to be sorted by report output within Sales Rep (rather than by Sales Rep within report output).

The **After_Burst_Batch** command line argument allows you to insert batch file operations **after each successful export burst step** (and waiting for that batch file to complete processing).

You can embed within the command lines inside the batch file field/formula names just as you can within the Visual CUT 3rd tab options. Visual CUT substitutes the appropriate values (those associated with the current processing cycle) for these field/formula names before launching the batch file processing. **This allows synchronized interweaving of group bursting printouts from multiple reports without the limitations of subreports (subreports don't support different paper trays, different paper orientations, and nested subreports)**.

For example, using the following scheduling string (or batch file):

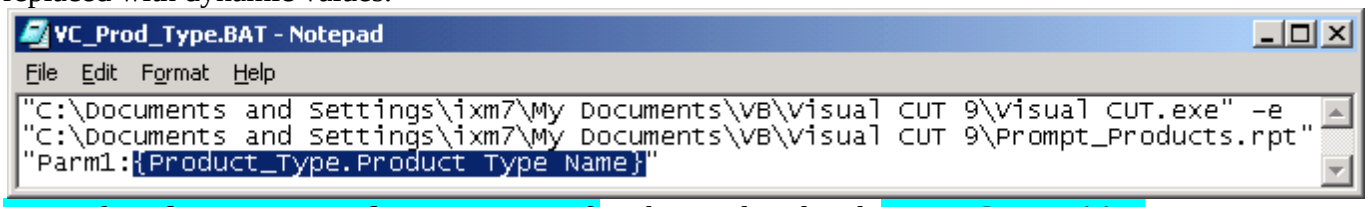
"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt" "Printer_Burst_Only:\\PS\Laser1"
"After_Burst_Batch:C:\VC_Prod_Type.bat"

---

would cause Visual CUT to:

1. Start processing the Visual CUT.rpt and burst the first group value ("Competition")

2. Before bursting the next group value ("Mountain"), Visual CUT looks into the **VC_Prod_Type.bat** file to see if recognized fields/formula names should be temporarily replaced with dynamic values.


VC_Prod_Type.BAT - Notepad
File Edit Format Help
"C:\Documents and Settings\ixm7\My Documents\VB\Visual CUT 9\Visual CUT.exe" -e
"C:\Documents and Settings\ixm7\My Documents\VB\Visual CUT 9\Prompt_Products.rpt"
"Parm1:{Product_Type.Product Type Name}"

Parm1:**{Product_Type.Product Type Name}** is then replaced with Parm1:**Competition**

3. The dynamic/temporary copy of the batch file is then triggered for processing, causing a secondary instance of Visual CUT to process the After-Bursting request.

4. Visual CUT then resumes the cycle with the next group value ("Mountain")

NOTE: in cases where paper tray choices within the reports don't seem to be honored during the processing, I recommend you **add the same Printer to your Windows setup a 2nd time with a different name and a different paper tray as the default**. Then set each report to use the printer "clone" that has the desired tray as the default.

Printer Job Name Functionality

When printing reports, Visual CUT names the print job according to the export file name option (even when only printing, and no exporting, takes place). Since the export file name can be a mixture of static text and field/formula values, this allows you to have a very fine control over the printer job names showing up in the printer job queue. For example, during print bursting, you can have each group printed under a different job name, allowing you to abort printouts for certain groups.

In cases where you don't have permissions to rename jobs in the printer queue, you should disable this functionality by setting the following option in the DataLink Viewer.ini to False:
[Options]
Rename_Printer_Jobs=FALSE

ZIP and Password Protect Files

You can instruct Visual CUT to ZIP and, if you provide a password, password protect any number of files. This can be particularly useful when emailing files that need to be protected or combined into a single attachment. The **zip processing occurs before emailing processing**. The command line argument structure is as follows:

… **"ZIP_Files:File_List>Password>Zip_File"**

The parameters (after the ":") are separated by a ">" and are as follows:

1. **File_List:** comma separated list of the files you wish to zip.
   If all files share the same folder, **you can specify the full path just for the first file**.
   If a file is not found, a warning is written to Failure.log and that file is skipped.
   Note: you can specify file names using **wild cards**.

2. **Password:** leave this option blank if you don't need to password protect the zip file. When avoiding password protection, the command line look like this:
   … **"ZIP_Files:File_List>>Zip_File"**

3. **Zip_File:** The path and name of the target zip file. If the specified path doesn't exist, Visual CUT would create it on the fly.

Important Note: you can use field or formula names within the command line arguments. The dynamic content of these fields/formulas would be substituted into the command line. Among other things, this allows you to **easily zip and protect file exports with different passwords for each group you are bursting.** For example (all in one line):

```
"ZIP_Files:c:\temp\{@Product} Sales.pdf,{@Product} Returns.pdf>
                {@Password}>c:\temp\{@Product} Reports.zip"
```

Load ini Values into Parameters

By naming parameters in a certain way, you can ask Visual CUT to automatically load their values from ini file entries.

One possible use may be for **reports where some parameters change only once per quarter**. Another case could be a situation where you develop and sell reports (probably .rpt files converted to .rpz files) to clients in a vertical market. These reports are designed to work against a known database structure, but **each client may need to customize the reports with text elements, conditional formatting, or record selection criteria without changing the report design**. Or perhaps you wish to **restrict use of your reports to only paying customers by providing a license code** that must match (using secret logic) the company name in the customer's database.

Instructions:

First, add to the [Options] section in **DataLink_Viewer.ini** (in the application folder) an entry with a **key name** (Company) and value (Millet Software) of your choice. For example:

```
[Options]
 …
Company=Millet Software
```

Then, add a **String Single-Value parameter** named **"DLV_INI_Option_KeyName"** to the report. **Visual CUT** automatically sets the value of such parameters to the value found for the Key Name under the [Options] section of DataLink_Viewer.ini. So, in the case above, the parameter value of **DLV_INI_Option_Company** would be set to "Millet Software".

Notes:

- The user never gets prompted **for the value of such parameters.**
- If a matching key name can't be found, the parameter gets the value of: "Failed INI Lookup"
- You can have as many parameters like this as you wish, each with a different key name.
- To pass such parameters to subreports, create a formula in the main report that simply returns the value of the parameter. Then, pass that formula as a link to the subreport.

Securing Reports against Unauthorized Use

If you wish to secure your reports against unauthorized use, you can provide authorized users a License Key string for the ini entry. Within the report (later distributed to clients as an .rpz file) you design a record selection criterion that returns true only if the license key matches a condition, such as the company name in the database. As a simple example, you could check the number of characters in the license key is equal to the length of the actual company name (in the database) plus the number of R's in that company's name.

FTP/SFTP

Exporting to an FTP Server (old approach)

One option for exporting to an ftp server is to map the target FTP folder into a "drive" letter on the machine where Visual CUT is running. From that point on, Visual CUT can simply export to that drive as if it's exporting to a local drive.

In cases where due to security or operating system restriction the mapping can't be done using Windows, you can use specialized utilities that take care of this mapping.

One such utility is free: ***FTPDrive***: http://www.killprog.com/fdrve.html

Another is ***WebDrive*** ($69.95): http://www.southrivertech.com/products/webdrive/index.html

Note: if you are exporting a PDF file to an FTP drive, and you don't need the extra PDF file properties and bookmark functionality provided by Visual CUT, consider turning that option off (in the Options dialog within Visual CUT). This functionality requires Visual CUT to reach out and handle the exported PDF file. In the case where that file sits on an FTP drive, this can slow down processing. Alternatively, you can use 2 command lines in the batch file: the first to export to the local hard drive, and the 2nd line that simply copies the resulting file to the FTP destination.

Uploading to an FTP Server (new approach)

You can instruct Visual CUT to FTP files to a web server. This can be particularly useful when you wish to export reports but email only a link to the exported file (avoiding the need to attach the export file).

The FTP Upload process occurs **before emailing but after exporting** and post-export processing. This allows you, for example, to merge pdf files, add bookmarks, table of content, and page numbers, and then upload the final pdf file so users can access it via the web.

The command line argument structure is as follows:
…**"FTP_Upload:Mode>>Server>>User>>Password>>file(s)>>directory"**

The parameters (after the ":") are separated by a ">>" and are as follows:

1. **FTP Connection Mode**. Possible Values are:
   a. **Passive_1**
   b. **Active_1**
   c. **AUTH_SSL/TLS_1**
      Visual CUT will detect server options and automatically selects SSL 3.0 or TLS 1.0 known as "AUTH SSL" or "AUTH TLS"
   d. **FTPS_1** (known as "Implicit SSL")

2. **FTP Server**: the Name or IP address of the FTP host

3. **User _ID** for authenticating to the ftp server

4. **Password** for authenticating to the ftp server (can reference Encrypted Password Name)

5. **File_List:** comma separated list of the files (**path & name**) you wish to upload. You can specify file names using **wild cards**, and Visual CUT will upload all matching files.

6. **Target directory:** The destination folder for the uploaded file(s).
   For example : **Sales/{@Year}/{@Product}**
   If some target folder levels don't exist they are created by Visual CUT.

7. **Maximum_Age_In_Minutes** (optional). If specified, older files are skipped. Useful when specifying files via wild card expressions and targeting newly created files.

As always, **you can use report field/formula names within the command line argument**. The dynamic content of these fields/formulas would be substituted into the argument.
For example:
…"**FTP_Upload**:Passive_1>>74.220.207.67>>user>>pass>>**{@Export_File}**>>/public_html/Download"
or
…"**FTP_Upload**:Passive_1>>74.220.207.67>>user>>pass>>**{@Export_File}**>>/public_html/Download>>**10**"

Uploading to an SFTP Server

You can instruct Visual CUT to upload files via the **SFTP** protocol using **SFTP_Upload** command line argument. SFTP and FTP are two completely different protocols.
Don't confuse FTP_Upload (described in the prior page) with SFTP_Upload.

The SFTP_Upload process occurs before emailing but after exporting and post-export processing. This allows you, for example, to merge pdf files, add bookmarks, table of content, and page numbers, and then upload the final pdf file so users can access it via the web.

The command line argument structure is as follows:
…**"SFTP_Upload:Arguments"**

The arguments (after the ":") are separated by a ">>" and are as follows:

1. **FTP Port**. Typically **22**
2. **FTP Server**: the **Name** or **IP address** of the SFTP host

3. **Authentication Mode: PW** (Password), **PK** (Private Key), **PWPK** (Both)

4. **User _ID** for authenticating to the server
5. **Password** for authenticating to server (leave blank if Authentication Mode is **PK**)

6. **Private Key File** e.g., **MyPrivateKey.pem** (leave blank if Authentication Mode is **PW**)
7. **Private Key File Password** needed only for encrypted Private Key Files. Blank otherwise

8. **File_List:** comma separated list of the files (**path & name**) you wish to upload. You can specify file names using **wild cards**, and Visual CUT will upload all matching files.

9. **Target directory (**can be blank**)** The destination folder for the uploaded file(s).
   Specified as absolute path or as relative to the user's account home directory.
   If this argument is left blank, the file would be placed in the **user's account home directory**.
   If some target folder levels don't exist they are created by Visual CUT.

10. **Maximum_Age_In_Minutes** (can be blank). If specified, older files are skipped. Useful when specifying files via wild card expressions and targeting newly created files.

As always, **you can use report field/formula names within the command line argument**.
The dynamic content of these fields/formulas would be substituted into the argument.
For example:
…"**SFTP_Upload:22>>ms.com>>PW>>user1>>pass1>>>>>>{@Export_File}>>public_html/Download>>**"
or
"**SFTP_Upload:22>>ms.com>>PK>>user1>>>>c:\keys\ido_dsa>>>>{@Export_File}>>public_html/Download>>10**"

Downloading from an FTP Server

The FTP Download process occurs **before emailing but immediately after exporting**.
The command line argument structure is as follows:
…**"FTP_Download:Mode>>Server>>User>>Password>>file(s)>>directory"**

The parameters (after the ":") are separated by a ">>" and are as follows:

1. **FTP Connection Mode**. Possible Values are:
   a. **Passive_1**
   b. **Active_1**
   c. **AUTH_SSL/TLS_1**
      Visual CUT will detect server options and automatically selects SSL 3.0 or TLS 1.0 known as "AUTH SSL" or "AUTH TLS"
   d. **FTPS_1** (known as "Implicit SSL")

2. **FTP Server**: the Name or IP address of the FTP host

3. **User _ID** for authenticating to the ftp server

4. **Password** for authenticating to the ftp server (can reference Encrypted Password Name)

5. **File_List:** semi-colon (;) separated list of the files (**path & name**) you wish to upload. You can specify file names using **wild cards (*)**, and Visual CUT will download all matching files.

6. **Target directory:** The local destination folder for the downloaded file(s).
   For example : **c:\temp**
   If some target folder levels don't exist they are created by Visual CUT.

As always, **you can use report field/formula names within the command line argument**.
The dynamic content of these fields/formulas would be substituted into the argument.
For example:

This downloads 2 specific files (if source folder remains the same, can specify it only once):
"FTP_Download:Passive_1>>www.MilletSoftware.com>>milletso>>Encrypted_Password_SFTP>>**/public_html/Download/Sales/{@Year_Parameter}/Sales for Hybrid in 2004.pdf;Sales for Locks in 2004.pdf**>>c:\temp\FTP_Download"

This uses wild card (*) expression to download multiple files:
"FTP_Download:Passive_1>>www.MilletSoftware.com>>milletso>>Encrypted_Password_SFTP>>/public_html/Download/Sales/{@Year_Parameter}/***.pdf**>>c:\temp\FTP_Download"

Extract Files Stored in Database

Databases can store images, documents, and media in binary column data types. You may need to extract such files to the file system for further processing (merging, emailing, etc.). Visual CUT can automate this using the **SQL_Extract_Files** command line argument:

… **"SQL_Extract_Files:DSN>>User ID>>Password>>Column>>Folder>>File Name>>Extention>>SQL Statement>>Unzip?"**

The parameters (after the "**:**") are separated by a "**>>**" and are as follows:

1. **DSN:** The ODBC DSN pointing to the database. Doesn't have to match DSN used in report.
2. **User ID:** Leave blank if no user id is needed to connect.
3. **Password:** Leave blank if no password is needed. May refer to named encrypted password.
4. **Column**: column name (in the SQL result set) storing the file content.
5. **Folder**: Path to folder where file is extracted (VC creates missing folders on the fly).
   Note: can refer to a column name.
6. **File Name**: Name for extracted file. If this includes the path, leave the previous option blank.
   Note: can refer to a column name.
7. File Extension: leave blank if File Name above already contains extension.
   Note: can refer to a column name.
8. **SQL Statement:** the SQL statement to execute. Typically references Crystal fields/formulas. **If statement returns multiple rows, one file is extracted for each row**.
9. **Unzip?**: If '**True**' the binary content is assumed to be zipped, so it gets unzipped.
   Note: can refer to a column name.

For example, the following command line argument

```
… SQL_Extract_Files:DSN1>>>>>>St_Picture>>c:\temp\>>St_Name>>.png>>
Select * From Students Where Major='{@Major}'>>False"
```

Would trigger a SQL statement through the DSN1 ODBC DSN, without user id and password, and materialize all Student images stored in the **St_Picture** column into the **c:\temp** folder as **.png** files named according to the **St_Name** column.

The Folder, File Name, File Extension, and Unzip? Parameters **may refer to database column names in the result set**. This allows a result set with multiple rows to extract each file to a different name (or even to a different folder) and to unzip the content only when needed. Alternatively, they **may be static text or references to Crystal fields/formulas**.

As always, **you can use report field/formula names within the command line argument**. The dynamic content of these fields/formulas would be substituted into the argument. For example, the command line above is referring to the {@Major} formula.

File Location Functionality

Visual CUT uses several files to store user preferences, report processing options & information:

1. **Visual CUT.mdb** stores report processing options
2. **DataLink_Viewer.ini** stores general options and user preferences
3. **ReportList.txt** stores information about previously opened reports for use in the 1st tab grid
4. **ReportList.grd** stores grid style information (grouping, column visibility, font size, etc.)
5. **Failure.log** records processing failure information
6. **Visual_Cut.log** records email communications with the SMTP server

By default, these files are in the common application data folder. However, for security or other reasons you may direct Visual CUT to use a different folder location for these main files. To direct Visual CUT to use and maintain these files at a different folder, open DataLink_Viewer.ini in the application folder and add a line such as follows under the File_Locations section.

```
[File_Locations]
Main_Files_Folder=c:\Visual CUT\
```

The specified folder must exist and the user should have full permissions on it.
When Visual CUT starts, if it doesn't find Visual CUT.mdb, DataLink_Viewer.ini, ReportList.txt, and ReportList.grd in the specified folder, it will attempt to copy these files from the application folder to the specified folder.

To direct command line processing to use a different Main_Files_Folder, you can use a **command line argument** like this: … "**Main_Files_Folder**:some path to a folder"
This is useful in scenarios where a centralized scheduler triggers processing on behalf of multiple users who maintain settings in their own folders.

Automatic Handling of Write-Protected Application Folders

When a user doesn't have write permissions to the application folder (typical of Vista and Windows 7 machines), Visual CUT handles that scenario as follows:

On load, if the user can't write to the DataLink_Viewer.ini, and the ini file doesn't already have a "Main_Files_Folder" entry in the [File_Locations] section then use DataLink_Viewer.ini if found in one of the following locations:

- common appdata (ProgramData) folder (common to all users), under "MilletSoftware\VC_11\"
- local appdata folder (local to the user), under "MilletSoftware\VC_11\"
- roaming appdata folder (follows the user to other machines), under "MilletSoftware\VC_11\"

Otherwise, this is probably the first time VC was started on that machine so we::

a) create a common appdata folder (e.g., c:\ProgramData\MilletSoftware\VC_11\
b) copy the DataLink_Viewer.ini to that folder
c) set its Main_Files_Folder option to that path
d) copy all the main files to that folder
e) provide a message to the user indicating the main files have been redirected

Note: The version information dialog shows the paths to the main files.

Directing the Visual CUT database to Another DBMS

By default, Visual CUT uses Visual CUT.mdb (or Visual CUT.accdb) as its repository for report processing settings. However, **if you would like to use a more robust DBMS, particularly if you want multiple users to concurrently update report processing options in the same centralized database, you may redirect Visual CUT to use any OLE DB compliant database**.

First, you need to create or import 4 required tables into your database (Report_Opt, Login_Opt, Export_Opt, and Report_Export_Options). Contact MilletSoftware for help with this process. A backup of a SQL Server database is available for easy restore into your SQL Server instance.

Specifying a Connection String

Use the [Options] section in the DataLink_Viewer.ini to specify the connection string

Example 1 - connecting to SQL Server Express using NT Authentication:

Connection_String="Provider=SQLNCLI11;Server=Srv1\SQL2;Database=Visual CUT;Trusted_Connection=yes;"

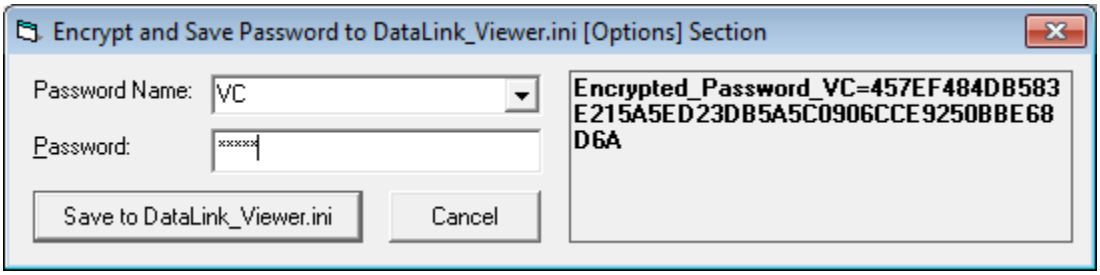
Example 2 - using SQL Server Authentication:

Connection_String="Provider=SQLNCLI11;Server=Srv1\SQL2;Database=Visual CUT;Uid=ido;Pwd=shhh;"

Example 3 - protecting password by referring to VC_Password_Encrypted

Connection_String="Provider=SQLNCLI11;Server=Srv1\SQL2;Database=VC;Uid=ido;Pwd=@VC_Password_Encrypted@;"

The ini file should have either a manually created entry named **VC_Password_Encrypted**, such as this:
VC_Password_Encrypted= F13210C40859455851BD7BE84880241A18BAFEE0E349C8ED
- or -
an entry called **Encrypted_Password_VC** generated via Options dialog, Process tab, **Encrypt & Save Password** button with a name of **VC**:
Encrypted_Password_VC=F13210C40859455851BD7BE84880241A18BAFEE0E349C8ED



Updating DataLink_Viewer.ini via a Delta File

You may want to automate the process of updating some of the DataLink_Viewer.ini settings. You can do that by placing a **DataLink_Viewer_Delta.ini** file in the application folder. Any entries found in that file will update DataLink_Viewer.ini when the application is launched.

Here is an example of a **DataLink_Viewer_Delta.ini** file:

```
[Delta_Options]
Delete_After=NEVER
//USE or NEVER or Some Date specified as yyyyMMdd
// if entry above not found, then default is USE
Update_Master_INI=False
// if entry above not found, then default  is False
Update_Slave_INI=True
// if entry above not found, then default is True

[Options]
Attempt_Logon_Without_Password=False
Strip_Table_Qualifiers=True
Saved_Data_Action=Display
Parameter_Values_Remember_Max_Chars=900
Saved_Parameter_Set_Minimum_N=5

[Integrated_Authentication]
Enable_Integrated_Authentication=True
```

The [Delta_Options] section is used only to control the following aspects of the process:

- The **Delete_After** option controls when the Delta ini file is deleted:
  - NEVER - the file will not be deleted
  - USE - the file is deleted after being used once
  - yyyyMMdd - the file is deleted after the specified date

- Update_Master_INI controls whether the Master ini file is updated.

- Update_Slave_INI controls whether the User ini file is updated.

Restricting User Actions

In some scenarios you may wish to restrict what the user is allowed to do with Visual CUT. For example, as an Administrator, you may want to set things up so users can run only some reports and process them only with saved settings.

You can elect to specify some or all of the following restricting options in the **[Options]** section of a **Master_DataLink_Viewer.ini** file **in the application folder**:

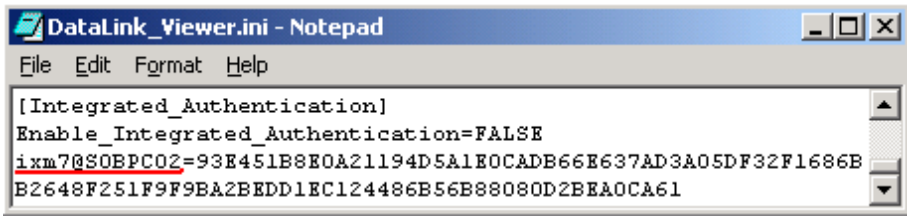
```
[Options]
// Disable Preview Buttons
Disable_Print_Button=TRUE
Disable_Export_Button=TRUE
Disable_Select_Button=TRUE
Disable_Search_Button=TRUE
Disable_Refresh_Button=TRUE

// Disable Visual CUT Buttons
Disable_Options_Dialog=TRUE
Disable_Browse_Dialog=TRUE
Disable_Version_Info=TRUE
Disable_Check_for_Updates=TRUE

// Make Processing Options Read Only
Processing_Options=Read_Only
// Disable the 'Log Email Activity' checkbox and hide the Notepad button to open the log
Disable_Email_Log_Activity_GUI=True
// Hide Email Queue (smtpQ) panels in the status bar
Disable_Email_Statusbar_GUI=True

// Disable certain Search & Replace categories
Disable_Find_and_Replace_Categories= Email_Addresses||Email_Information
```

Integrated Interactive Authentication

While command line processing relies on encrypted login information stored in Visual CUT.mdb, some users may also wish to avoid repeated logins during interactive use.

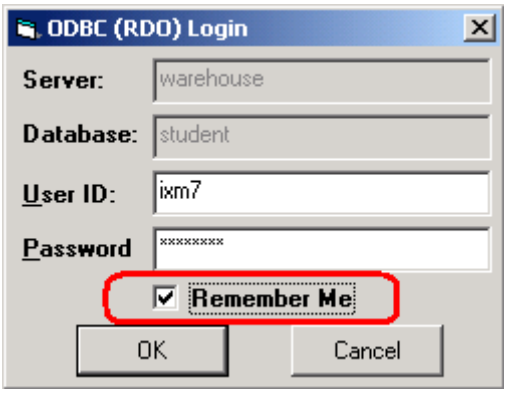
Integrated Authentication ("Remember Me")

To support interactive integrated authentication, the database login information is stored, highly encrypted, inside **DataLink_Viewer.ini** as shown below:



To ensure full security, this encrypted format includes an internal identification of which **Windows User & PC** this login information belongs to. **Visual CUT uses this encrypted database login information only after checking that the same Windows user is running from the same PC.** In other words, users cannot break the login security by attempting to copy and paste the encrypted information to their own ini file entry).

For example, in the example shown above integrated authentication has been enabled. This can be done via a checkbox in the Option dialog (Process Tab). The user (**ixm7**) then logged in to a secure database by providing a **database user id & password**. The user turned on the "**Rememebr Me**" option (visible only when integrated authentication is enabled):



The information was then saved for the **ixm7** user, running on the **SOBPC02** PC as shown in the ini file above. **From that point on, the same user (ixm7), once logged to the same PC (SOBPC02), doesn't need to manually login to the same data source.**

The same user can turn on the "Remember Me" option for **unlimited number of secure data sources** and the information for all of them would be maintained inside the encrypted entry. The Options dialog allows each user to delete their own integrated authentication information by clicking a button.

Shared Machine Authentication

This section describes how one user can elect to share their integrated authentication information with any other user who successfully logged in to the same machine.

Step 1: First, add the entry in bold to the DataLink_Viewer.ini file:

[Integrated_Authentication]
Enable_Integrated_Authentication=TRUE
ixm7@SOBPC02=18CB9CAE3D8BF3484000123301DD155638EAD4AD6B4D622C04DF125B7404BCBC4717A5DA2FBCD94314A7CE63BBF7357E
Enable_Shared_Machine_Authentication=TRUE

Step 2: Then, as the "key" user who will share integrated authentication, run a report and get to a login dialog. Note: if you already have integrated authentication for yourself, use Forced Login. Alternatively, discard your integrated authentication (using the Trash Can button in the Options dialog). Make sure you turn on the Remember Me option in the login dialog, just like setting up integrated authentication for yourself.

Step 3: Because of the **Enable_Shared_Machine_Authentication=TRUE**
you would get a dialog asking you if you wish to share your integrated authentication functionality with other users who logged in to the same machine. This ensures a user can't be tricked into sharing integrated information without their knowledge and expressed consent. Click YES. If you then open the DataLink_Viewer.ini file you would notice this process generated a new entry (in bold):

[Integrated_Authentication]
Enable_Integrated_Authentication=TRUE
ixm7@SOBPC02=18CB9CAE3D8BF3484000123301DD155638EAD4AD6B4D622C04DF125B7404BCBC4717A5DA2FBCD94314A7CE63BBF7357E
Enable_Shared_Machine_Authentication=TRUE
Shared_Machine_Authentication=C8B6CD373D5D5ADDC6FE9F379521C7318FA297CB595B992C

That entry, in my particular case, points to the **ixm7@SOBPC02** integrated authentication entry.

Notes:

1. **There can be only one shared machine authentication entry**. If you need to change to another administrator, delete the **Shared_Machine_Authentication** line and let the new administrator go through the Remember Me and dialog step.

2. If the administrator adds data sources and login information to their integrated authentication information, all the machine users would have access to the new data sources. This is because the entry is really a "pointer" to whatever that administrator has accumulated in their entry.

3. If you interactively switch between DSNs and you want Integrated Authentication functionality, set **Enable_Integrated_Authentication_For_DSN_Changes** to **True**
This is a rare scenario so, for more detail, contact Millet Software.

Auto-Refreshing Web Dashboards

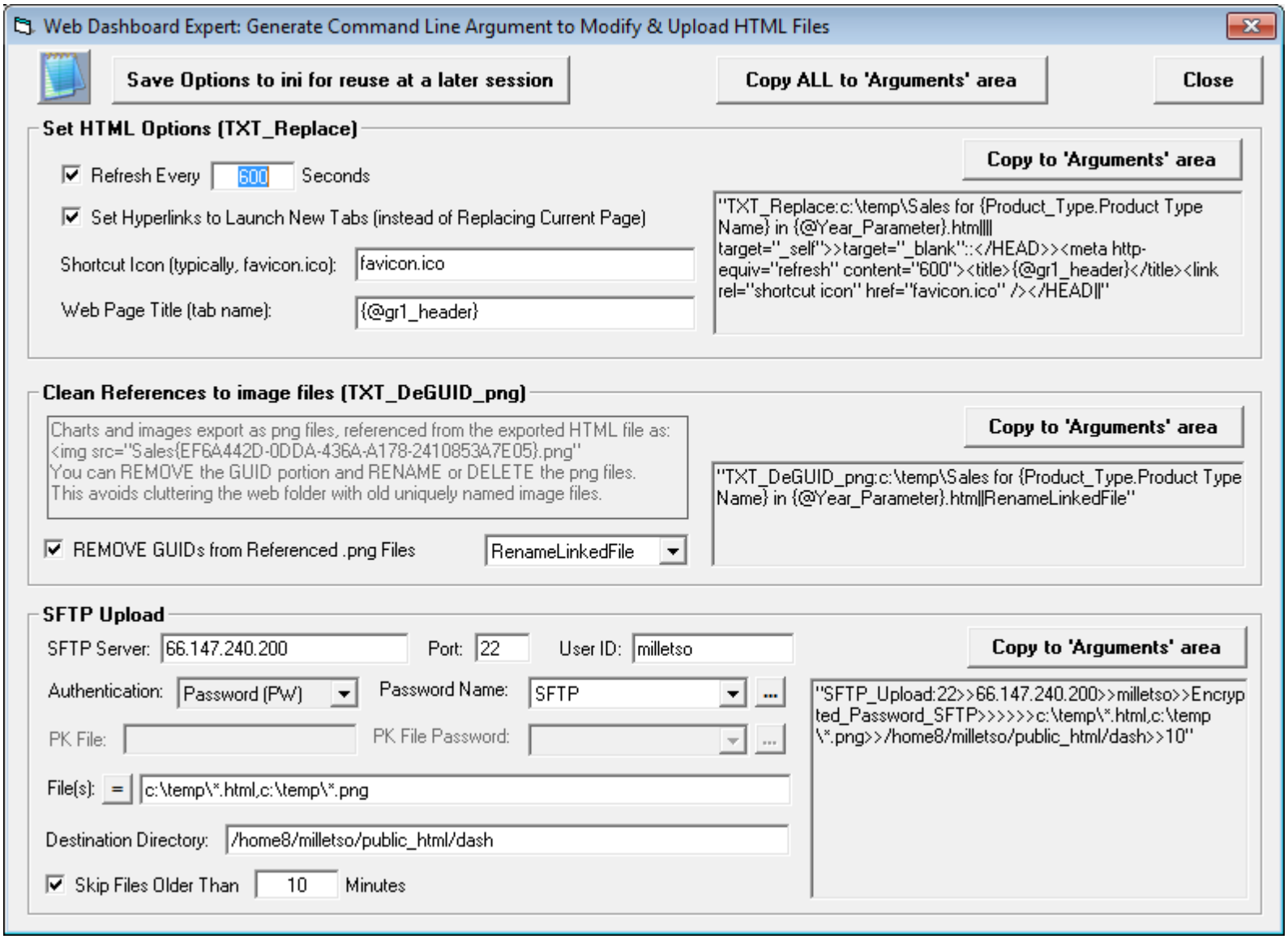
You can use Visual CUT to easily implement auto-refreshing dashboards on a web server. This allows users with only a web browser to monitor relevant information.

Web Dashboard Expert

When exporting to HTML 40, the options button ... launches a window allowing you to set various options for post-processing the HTML file. These options include: a) adding an auto-refresh behavior, b) Setting hyperlinks to launch to new tabs, c) Specifying tab titles and icons, d) Removing GUIDs from referenced.png image files, and e) SFTP Uploading HTML and Other files to the web server.

The window generates command line arguments that you can copy to the 'Arguments' area in the Export/Email tab. These options are then be applied during interactive/scheduled processing.

A web dashboard using that approach is available at:
http://academicweb.psu-erie.bd.psu.edu/laboccupancy/Labuse.html


Web Dashboard Expert: Generate Command Line Argument to Modify & Upload HTML Files
Save Options to ini for reuse at a later session
Copy ALL to 'Arguments' area
Close
Set HTML Options (TXT_Replace)
Refresh Every 600 Seconds
Set Hyperlinks to Launch New Tabs (instead of Replacing Current Page)
Shortcut Icon (typically, favicon.ico): favicon.ico
Web Page Title (tab name): {@gr1_header}
Copy to 'Arguments' area
"TXT_Replace:c:\temp\Sales for {Product_Type.Product Type Name} in {@Year_Parameter}.html|||| target="_self">>target="_blank"::</HEAD>><meta http-equiv="refresh" content="600"><title>{@gr1_header}</title><link rel="shortcut icon" href="favicon.ico" /></HEAD||"
Clean References to image files (TXT_DeGUID_png)
Charts and images export as png files, referenced from the exported HTML file as:
<img src="Sales{EF6A442D-0DDA-436A-A178-2410853A7E05}.png"
You can REMOVE the GUID portion and RENAME or DELETE the png files.
This avoids cluttering the web folder with old uniquely named image files.
Copy to 'Arguments' area
"TXT_DeGUID_png:c:\temp\Sales for {Product_Type.Product Type Name} in {@Year_Parameter}.html||RenameLinkedFile"
REMOVE GUIDs from Referenced .png Files
RenameLinkedFile
SFTP Upload
SFTP Server: 66.147.240.200
Port: 22
User ID: milletso
Copy to 'Arguments' area
Authentication: Password (PW)
Password Name: SFTP
PK File:
PK File Password:
File(s): c:\temp*.html,c:\temp*.png
Destination Directory: /home8/milletso/public_html/dash
"SFTP_Upload:22>>66.147.240.200>>milletso>>Encrypted_Password_SFTP>>>>>>c:\temp*.html,c:\temp*.png>>/home8/milletso/public_html/dash>>10"
Skip Files Older Than 10 Minutes

IFrame Approach

Alternatively, Visual CUT can export a report to HTML "display" page every N seconds. A "dashboard" web page, visited by the users web browser:
a) reloads itself every N seconds using an HTML tag such as:

<META http-equiv=refresh content=30>

b) loads the html "display" page into an IFrame areas (think about an IFRame as a picture frame for displaying another HTML page).

Using IFrames allows the hosting page to display multiple reports and use various GUI elements (Menus, Tabs, Accordions, and Data Tables). A sample web dashboard using that approach is available at: http://www.milletsoftware.com/dash/dash_demo_tabs.html
user id & password: **demo**

Visual CUT keeps refreshing the HTML display page by exporting to it every N seconds, and the dashboard page keeps refreshing itself and reloading the HTML display page. Hence, the user experiences an auto-refreshing dashboard by simply having that page open in a browser.

Here is what the dashboard HTML page may look like:

```
<HTML><HEAD><TITLE>Backlog Dashboard</TITLE>
<META http-equiv=Content-Type content="text/html; charset=windows-1252">
<META http-equiv=refresh content=30>
<META content="MSHTML 6.00.2900.3314" name=GENERATOR></HEAD>
<BODY>
<P><IFRAME border=0 name=I1
src="E:\Dashboards\Dashboard_1.html"
frameBorder=0 width=1048 height=20000>
</body>
</html></IFRAME></P></BODY></HTML>
```

Preventing File Locking

In order to prevent any chance of file locking, Visual CUT can export the reports to one folder and then copy (or FTP) the resulting HTML file to the web folder. Here is an example of a simple batch file that executes the Visual CUT export and then copies the resulting HTML page:

```
"C:\Program Files\Visual CUT 11\Visual CUT.exe" -E "c:\Reports\Dash1.rpt"
Copy c:\temp\Dashboard_1.html E:\Dashboards\Dashboard_1.html
```

Case Study

Phillip Scheel, a data analyst at CPS Energy, was kind enough to provide a pdf file describing the approach and several dashboards he has developed.

Hyperlinks

To allow users to click and open other web pages or any other document type you can assign to any report object within Crystal a hyperlink expression.

A typical approach is to have Visual CUT export a dashboard report, say a summary report of Sales By Product Type, to one HTML page as the main web dashboard. The batch file triggering the process can have another command line to trigger bursting of another report, creating one HTML page showing Sales by Product within each Product Type:
Detail_ for_{Product_Type.Product Type Name}.html
You would then set the hyperlink expression for the Product Type field in the main dashboard report to dynamically point to its related detailed HTML page. The user can then click on each Product Type field in the main dashboard to drill-down into the detailed information in the bursted HTML pages. In a similar manner, drill-downs to related PDF or Excel files can be provided.

Note: use the Web Dashboard Expert to change hyperlink properties so that they open related web pages in a separate Window rather than replacing the content in the dashboard frame.

Tooltips

To provide users with related information when hovering over elements in the web dashboard, simply specify a **dynamic tooltip expression** for these objects in Crystal.
By inserting a line break character in these expressions, as in:

```
{@Cust} +  Chr(10) + {@Cust_Info}
```

you can present that content as **multiple lines**
note: Firefox seems to be the only browser that doesn't honor such line breaks.

Generating HTML via Email Message Body

Instead of relying on Crystal's HTML export, you may take advantage of the email message HTML editor and Visual CUT's ability to embed dynamic values from report fields/formulas in the message body (see video demo).

You can instruct Visual CUT to save the email message body to an HTML file using the ***Email_Message_Save*** command line argument. You can also skip the actual emailing by specifying "**VC_Skip_Email**" in the **Email_To** option.

This technique provides you with full control over the HTML being generated. As shown in the video demo above, it allows tables in your web dashboard to adjust to the browser screen size. It also allows you to directly add CSS and JavaScript to the generated HTML page.

Cleaning png File References

When the exported report contains images, Crystal generates uniquely named (using GUIDs) png files, which are referenced from within the HTML file. In many cases, you may wish to "clean" these file references by removing the GUIDs from the file references and renaming the png files. Visual CUT provides two command line arguments to achieve this: **TXT_DeGUID_png** and **TXT_Replace_Tokens**. See the following user manual sections for more detail:

- Removing GUIDs from png Files Referenced in HTML Exports
- Replacing Content in Text/HTML Files – Token Approach

Restricting Access to Web Dashboards

If you wish to restrict access to your web dashboards, a simple and secure method is described here: http://www.javascriptkit.com/howto/htaccess3.shtml

PDF Functionality

Creating Bookmarks (Group Tree) in Exported PDF Files [Old Approach]

Visual CUT can create a "Group Tree" of bookmark inside exported pdf document.
It allows users to easily navigate through large PDF files.

You specify the Bookmark label & page by inserting a **string formula in each Group Header that contributes a level to the "Group Tree." The formula/section can be suppressed.**

Naming the Bookmark Formulas:
For Group Header Level **1**, the name of the formula must be **VC_pdf_bookmark1** , …
For Group Header Level **5**, the name of the formula must be **VC_pdf_bookmark5**
Note: you can add other formulas at the same level (in the same section or split section) by adding a digit between 1 and 9. For example, if you used **VC_pdf_bookmark2** for GH2a you can use **VC_pdf_bookmark21** in GH2b and **VC_pdf_bookmark22** in GH2c.

Structuring the Formulas:
The formulas must return a **string** structured as the **Bookmark Label**, followed by "**::**" and ending with the **Page Number** where the formula was evaluated.

The sample report (**Visual CUT.rpt**) demonstrates this functionality:

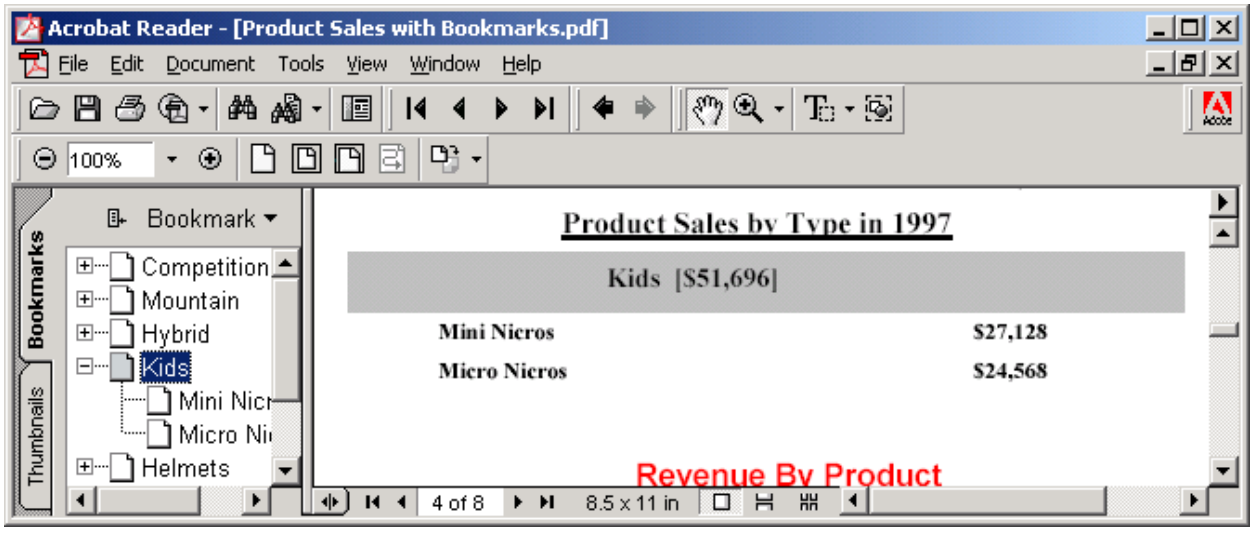
- On Group Header **1** the **VC_pdf_bookmark1** formula is defined as:

```
IF InRepeatedGroupHeader THEN ""
  ELSE {Product_Type.Product Type Name} + "::" + Cstr(PageNumber,0,"")
```

- On Group Header **2** the **VC_pdf_bookmark2** formula is defined as:

```
IF InRepeatedGroupHeader THEN ""
  ELSE GroupName ({Product.Product Name}) + "::" + Cstr(PageNumber,0,"")
```

After a PDF export, the resulting file has a 2-level hierarchy of Bookmarks and **clicking on any bookmark takes you to the page associated with that bookmark**:



Controlling PDF Bookmark Colors (& Text)

If you include (anywhere) in the Bookmark Label the color constants of **crRed**, **crBlue** or **crGreen** (note: case sensitive) Visual CUT would set the color of the bookmark text accordingly (and remove the color constant text from the label).
This functionality is useful for indicating item performance or status visually (e.g., red color indicating poor performance). Note: remember that since you control (via a formula) the bookmark text, you can also add various label indicators such as **[+]**, **[~]**, **[!]**, or **[-]**.

The level 2 bookmark formula (VC_pdf_bookmark2) in the sample report demonstrates this approach as follows:

```
WhilePrintingRecords;
IF InRepeatedGroupHeader THEN ""
ELSE
(
IF Average ({@Days_To_Ship}, {Product.Product Name}) > 5.0 THEN
"[!]crRed " + GroupName ({Product.Product Name}) + "::" +
Cstr(PageNumber,0,"")
ELSE IF Average ({@Days_To_Ship}, {Product.Product Name}) < 3.0 THEN
"[+]crGreen " + GroupName ({Product.Product Name}) + "::" +
Cstr(PageNumber,0,"")
ELSE
"[~]crBlue " + GroupName ({Product.Product Name}) + "::" +
Cstr(PageNumber,0,"")
);
```

Controlling How Many Bookmark Levels Are Initially Expanded

By default, the resulting PDF file will show only 1 (the top) level of bookmarks expanded, so that level-2 bookmarks (if there are any) are visible. Obviously, users can expand/collapse the bookmarks by clicking on the + or – nodes within the Adobe Acrobat reader. However, you may wish to present the users with a pdf where only the top-level bookmarks are initially visible or where more levels are initially expanded.

To change the default number of expanded levels for all processed reports, change the value of the PDF_Bookmarks_Open_Levels option in the DataLink_Viewer.ini
For example:

```
[Options]
PDF_Bookmarks_Open_Levels=0
```

To override the default number of expanded levels when processing a specific report, use the PDF_Bookmarks_Open_Levels command line argument as follows:

… "PDF_Bookmarks_Open_Levels:N"

Where N is the number of levels to be initially expanded.
For example, to have only the top bookmarks visible, use:
… "PDF_Bookmarks_Open_Levels:**0**"

Guarding Against Null Bookmark Values

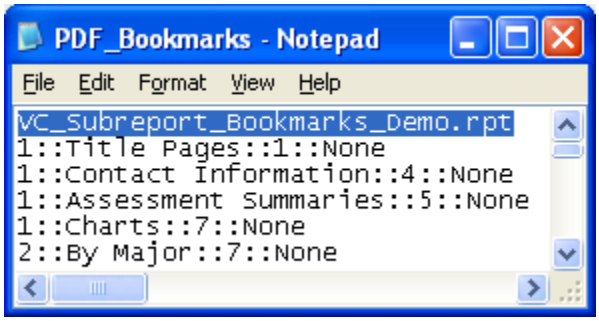
If the field(s) used to control the bookmark label is Null, the whole formula would result in a Null value and bookmark processing would be compromised. If this is a possibility given your data you should guard against it using a formula like this:

```
WhilePrintingRecords;
IF InRepeatedGroupHeader THEN
""
ELSE
IF IsNull({Product_Type.Product Type Name}) THEN
" " + "::" + Cstr(PageNumber,0,"")
ELSE
{Product_Type.Product Type Name} + "::" + Cstr(PageNumber,0,"");
```

Generating PDF Bookmarks from within Subreports

As an alternative to generating bookmark information from Group Header sections in a main report (as described above), Visual CUT allows you to specify bookmarks for a given PDF file by providing a text file (**PDF_Bookmarks.txt**) containing the bookmark information. This file must be located in the Visual CUT application folder and must be structured as follows:

1. The 1st line must match the RPT file name of the report being processed
2. The following lines provide 4 data elements separated by a "**::**" delimiter:
   a. Bookmark Level
   b. Bookmark Label
   c. Page Number
   d. Group Level 1 Value (used only in bursting operations)



Using "CUT Light" (a User Function Library described on my web site) or any other UFL that adds the ability to append text to text files from within Crystal formulas, **you can generate this text file automatically by embedding formulas in any report or subreport sections.**

Note that this provides functionality that goes beyond the Group Tree limitations of Crystal reports. Crystal can't generate Group Tree information based on information within subreports. However, **using this technique you can generate Group Tree information as PDF bookmarks based on information from both main as well as subreports!**

There are 2 main steps in the process:

1. Visual CUT runs the report and, because the report has special UFL formulas, the **PDF_Bookmarks.txt** file gets populated with bookmark information,
2. Visual CUT export the report to PDF, detects the presence of **PDF_Bookmarks.txt** and uses it to generate the bookmarks in the exported pdf file.

If you are interested in using this functionality, I can e-mail you a sample report (demonstrating the formulas needed to generate the text file information). I can also e-mail you download instructions for evaluating *CUT Light* (UFL allowing Crystal formulas to create and add data to text files).

Adding Bookmarks Using Crystal Formulas as Tags [New Approach]

Using a command line argument, you can instruct Visual CUT to look for invisible tags inside the pdf file (Crystal formulas with font color set to background color) and to use their content to generate bookmarks. Compared to the old approach (described in the previous section), this new technique has several advantages:

1. The bookmarks link to the **exact vertical location** (on the linked page) where the rendered formula is located. In the old approach, bookmarks only pointed to at the top of the page.
2. You can easily **generate bookmarks from within subreport and from any report section** (not just Group Headers). For example, you can generate bookmarks from Detail Sections or Group Footers.
3. You can **control the bookmark text color more completely** (instead of a few color constants, you can specify any color using **RGB** (Red/Green/Blue) values.
4. You can **control the expanded/collapsed state** of each bookmark node
5. You can **control the style** of the bookmark node (Regular, Bold, Italics, Bold&Italic)
6. **No need to worry about PageNumber resets, Keep Together, and WhilePrintingRecords properties that can cause headaches with the old approach**. This is because the new approach doesn't require specifying a page number and doesn't depend on the evaluation time of the bookmark formulas.
7. **You can avoid the need to use a command line argument** by setting **PDF_Bookmark_Tags_Default=True** under the [Options] section of DataLink_Viewer.ini (only applies to files < 100MB) Visual CUT would then process bookmark tags within an exported PDF file.

Here's an example of the command line argument structure:
… **"PDF_Bookmark_Tags:c:\temp\Sales in {@Year_Parameter}.pdf"** **Or**
… **"PDF_Bookmark_Tags:Source_pdf_file>target_pdf_file"**

If only one pdf file is specified (as in top example), the source file becomes also the target file.

The Crystal formulas act as tags for controlling the level, vertical location, text, color, expand/collapse status, and style (regular/bold/italic) of the desired pdf bookmarks.

Setting Up a Crystal Report with pdf formula tags

You can download a sample report demonstrating the technique from:
www.milletsoftware.com/Download/Visual_CUT_PDF_Bookmark_Tags.rpt
And the resulting pdf file from:
www.milletsoftware.com/Download/Visual_CUT_PDF_Bookmark_Tags.pdf

The sample report uses 3 formulas to generate bookmarks:
{@Bookmark_L1} is placed in the Group Header level 1 to generate **Product Type** Name bookmarks with bold blue text. These level 1 nodes are expanded.

{@Bookmark_L2} is placed in the Group Header level 2 to generate **Product Name** bookmarks with bold text. These level 1 nodes are collapsed
.

{@Bookmark_L3} is placed inside the **subreport** in the Group Header level 1 to generate **Customer Name**. Bookmarks with text colored in different shades of red or green depending on the average time it takes to ship orders of that product to the customer.

The generated bookmarks link not only to the page where the formula was rendered, but also to the exact vertical position on that page (minus 20 millimeters to provide some margin). The bookmarks are created for each rendered instance of the formula. For example, if you place the formula in a Group Header 1 you will generate a bookmark for each Group at level 1.

The formula text must be rendered within the formula boundaries in a single line. Use small font sizes: 2 or even 1 to achieve this. Then, turn the font color to White (or same color as the background) to make the formula invisible or, keep the **PDF_Tags_Delete_Default** option in DataLink_Viewer.ini as **True** and Visual CUT would remove the tag text after processing it. See box below for Sample formula.

Notes:

1. As always, you can use field or formula names within the command line argument. The dynamic content of these fields/formulas would be substituted into the argument. For example, {@Bookmark_L3} (inside the subreport) dynamically sets the color of the bookmark based on the performance of records in that node. Similarly, you may dynamically expand the bookmarks for groups with extreme performance measure to focus the manager's attention on those areas.
2. Visual CUT can remove pdf processing tags from the pdf file after processing those tags. This is controlled by an entry called **PDF_Tags_Delete_Default** under the [Options] section of DataLink_Viewer.ini. By default, this option is set to True. To ensure all tags are removed, use the Replace() function in Crystal to change double quotes into single. For example, the expression for the title in PDF_Bookmark_Tags could be:
   Replace({Customer.Customer Name},"""","'")
3. In Crystal, **use non-proportional font** for the tag formula (avoid **Calibri** and **Bold/Italic**)

Here is the commented {@Bookmark_L2} formula from the sample report at:
www.milletsoftware.com/Download/Visual_CUT_PDF_Bookmark_Tags.rpt

```
// Make the font size very small (2 or even 1 point) so content fits in a SINGLE line.
// The vertical location of the tag (minus default margin of 20 millimeters) controls the vertical
//     location of bookmark page target.  You can override the margin using an optional argument.
// The formula always starts with "#BM_Tag::" and ends with "#"
"#BM_Tag::" +
"2" +  "||" +  // Bookmark Level. During bursting, levels are shifted up automatically
Replace({Product.Product Name},"""","'") + "||" +  // Bookmark Text (" >> ')
"-" + "||" +  //"Expand Status:  + to open the bookmark or - to collapse it.
"1" + "||" + // Text Style:  0=regular  1=Bold   2=Italic   3=Italic Bold
"0;0;0" +  // Color as RGB:  3 numbers between 0 and 255 separated by ";"
// "||10" + // [Optional] Vertical Margin.  Leave commented or uncomment to
            //    override the default of 20 millimeters.
"#"
```

Adding a Table of Content to a PDF File

You can generate a Table of Contents (TOC) inside a PDF file based on its Bookmarks. Bookmarks are used during online viewing – a TOC is more useful for a printed document. The command line argument structure is as follows:

… **"PDF_TOC:1>40>11>6>5>2>pdf_file_path_and_name"**

The parameters (after the ":") are separated by a ">" and are as follows:
1. **Page** where the TOC is inserted. **1** is a typical choice. Specify 999999 to insert as last page(s).
Note: you can specify two numbers separated by two vertical bars in cases where you want to offset the page numbers shown in the TOC. For example, **2||1** would insert the TOC in page 2 (after a title page perhaps) but refer to page #2 as page #1 and apply a similar offset to all other page numbers that are displayed in the TOC.
2. The right (& left) **page margin** (in millimeters). **40** is a typical choice
3. The **font size** (in points). **11** or **12** are typical choices.
4. The **row spacing** (in millimeters). **6** is a typical choice.
5. The **indentation** (in millimeters) added to each hierarchy level. **5** is a typical choice.
6. The **Minimum Bookmark Level** below which bookmarks are ignored. **2** is a typical choice.
7. OPTIONAL: The PDF file path & name (for example, **c:\temp\other_file.pdf**).
   Leaving this argument blank, would default processing to the file being exported.
   Providing a file name would direct processing to the specified file (even if it's not the exported PDF file.

Overriding the default "Table of Contents" Header Text
You can control the "Table of Contents" header text by adding/updating the [Text_Change] section in the DataLink_Viewer.ini file. For example:

```
[Text_Change]
Table of Contents=Table des matières
Table of Contents (cntd)=Table des matières (suite)
```

If you enter no text after the equal sign, Visual CUT will suppress the header text.

A "Table of Contents" (or the equivalent text as per the section above) is set to the 1st page where the Table of Contents was inserted (regardless of the page number specified by you).

Visual CUT assigns the top level of the TOC a bold font and a slightly larger row spacing and assigns level 3 and below a smaller font and a smaller row spacing.

If the TOC requires more than 1 page, Visual CUT inserts additional pages for the TOC.

The TOC rows (just as Bookmarks) are linked to the specified page so clicking on a TOC row takes you to that page. Regardless of the number and location of the inserted TOC pages, all page links (bookmarks and TOC rows) are adjusted accordingly.

The process takes care of generating a bookmark for the Table of Contents.

Advanced Table of Content Options

You can add page header images and gain fine control of font type, font size, font color, row spacing, bullets, indents, page orientation, and vertical page margins for the Table of Content by specifying these options in a [PDF_TOC] section inside DataLink_Viewer.ini.

For example, the following section starts by specifying vertical page margins for the first and continued Table of Content pages. It then specifies the images (with web hot links) to be inserted at the top of these pages. The rest of the items control font type, size, color, bullets, spacing, and indentation for each level in the Table of Content hierarchy.

```
[PDF_TOC]
// the Page_Margin_Top entry is required to enable this functionality
Page_Margin_Top=50
Page_Margin_Top_Continued=30
Page_Margin_Bottom=30
Page_Orientation=Portrait

// elements are: Left>Top>Width>Height>web link>image file>border (0=no border, 1=border)
// Left, Top, Width and height are all in millimeters. Left & Top are relative to top corner of the page
Top_Image_1st=0>0>216>40>http://www.MilletSoftware.com>C:\temp\header.jpg>0
Top_Image_Continued=0>0>216>20>http://www.milletsoftware.com/Visual_CUT.htm>C:\temp\VC.jpg>0

// True Type font specified as TT||Font Name||point size||Style: Regular/Bold/Italic/BoldItalic||RGB color
Default_Font=TT||Garamond||11||Regular||0;0;0
// Standard font specified as:
// = ST||Font Name: Helvetica/Courier/TimesRoman||point size||Style: Regular/Bold/Italic/BoldItalic||RGB color

Header_Font=TT||Garamond||16||Bold||16;78;139

Level_1_Font=TT||Garamond||12||Bold||225;0;0
Level_1_Row_Spacing=8
Level_1_Bullet=
Level_1_Bullet_Spaces=
Level_1_Indent=

Level_2_Font=TT||Garamond||11||Regular||0;0;0
Level_2_Row_Spacing=6
Level_2_Bullet=149
Level_2_Bullet_Spaces=3
Level_2_Indent=1
```

You can see a sample pdf generated using these options at:
www.MilletSoftware.com\Download\VC_PDF_Table_of_Contents_Sample.pdf

To override the settings above for a specific report, create a similar section and call it: [PDF_TOC_YourReport.rpt]

Note: you can use field or formula names within the ini entry values, just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. The dynamic content of these fields/formulas would be substituted into the ini values. For example, this allows a different image to be used for the Table of Contents header depending on data in the report.

Adding Page Numbers to a PDF File

You can instruct Visual CUT to add page numbers to the exported PDF file. This is useful when also adding a Table of Contents (as described above) since in such a case the page numbers from Crystal would be wrong. The command line argument structure is as follows:
… **"PDF_PAGE_N:2>10>10>11>Bottom>Center>Page_NofM**>pdf_file>TimesBold**"**
or
… **"PDF_PAGE_N:2>10>10>11>Bottom>Center>Page_NofM**>pdf_file>**TT||Garamond||Bold||225;0;0"**

The parameters (after the ":") are separated by a ">" and are as follows:
1. **Staring with Page: 2** is used in cases where the 1st title page shouldn't be numbered.
* specify two numbers separated by || (N||M) to start page numbering on page N with the number M. For example, **2||1** would start numbering on page 2, but with the number 1.
* specify 3 numbers separated by || (N||M||X) to start page numbering on page N with the number M, but skip the last X pages. For example, **3||2||1** would start numbering on page **3**, with the number **2**, but skip **1** last page.

2. **Horizontal page margin** (in millimeters). Doesn't apply to Centered layouts. **10** is typical.

3. **Vertical page margin** (in millimeters). **10** is typical.

4. The **font size** (in points). **11** or **12** are typical.

5. The **vertical position** on the page (**Top** or **Bottom**). **Bottom** is typical.

6. The **horizontal position** on the page (**Left**, **Center**, or **Right**). **Center** is typical.

7. The format (**N**, **Page_N**, or **Page_NofM**). **Page_NofM** is typical (e.g., **Page 3 of 210**).

8. OPTIONAL: The **PDF file path & name** (for example, **c:\temp\other_file.pdf**).
   Leaving this argument blank, would default to processing the file being exported.
   Providing a file name would process the specified file.

9. OPTIONAL: The **font type** (default is Helvetica).
   Possible font types:
   Courier, CourierBold, CourierBoldItalic, CourierItalic
   Helvetica, HelveticaBold, HelveticaBoldItalic, HelveticaItalic
   TimesRoman, TimesBold, TimesItalic, TimesBoldItalic
   * or * any TrueType Font specified as 4 elements separated by || like this::
   TT||FontName||Style||Red;Green;Blue
   TT constant indicating this is a True Type font (target machine should have it installed)
   Font Name such as **Garamond**
   Style: **Regular**, **Bold**, **Italic**, or **BoldItalic**
   Color as RGB: 3 numbers between 0 and 255 separated by ";"
   Black=0;0;0 Maroon=225;0;0 etc. (see RGB #s at: http://web.njit.edu/~kevin/rgb.txt.html)
   For example: **TT||Garamond||Bold||225;0;0**

Adding Web/File/email Hotspot to a PDF File

You can instruct Visual CUT to add a rectangular hotspot to a pdf file that acts as a web browser link to a web URL, email address (mailto), or a file on your local machine.

The command line argument structure is as follows:

…**"PDF_LinkToWeb:c:\test.pdf>1>99>40>60>50>10>http://www.MilletSoftware.com>>>0"**

The parameters (after the ":") are separated by 10 ">" and are as follows:

1. **pdf file** to which the hotspot would be added.

2. **Staring with Page**

3. **Ending with Page**: use a very large number if you want to apply the hotspot to all pages and you don't know how many pages are in the document).

4. **Left** (in millimeters): The left edge of the hotspot rectangle (relative to top left corner of the page).

5. **Top** (in millimeters). The top edge of the hotspot rectangle (relative to top left corner of the page).

6. **Width** (in millimeters). The width of the hotspot rectangle

7. **Height** (in millimeters). The height of the hotspot rectangle.

8. **Link.** for example: **"http://www.MilletSoftware.com" or "mailto:ido@MilletSoftware.com"** or **"file:///c:\\temp\\helloworld.bat"**

9. **Text**: the text to add inside the hotspot rectangle (the text font size will be adjusted to automatically fit the text inside the rectangle area. Leave blank (as demonstrated above) if you don't want to add any text. Do not include the character '>' inside the text.

10. **Text Vertical Alignment**: Top, Center, or Bottom. Leave blank (as demonstrated above) if you don't want to add any text.

11. **Border**: 1 for visible border, 0 for no border (default for missing or invalid value is 1)

Notes:
You can have multiple PDF_LinkToWeb command line arguments in a single command line.

You can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. The dynamic content of these fields/formulas would be substituted into the command line argument. For example (all in one line):
…**"PDF_LinkToWeb:{@pdf_file}>1>99>40>60>50>10>{@Link}>{@Text}>Center>1"**

Adding an Images with an Optional Hotspot to a PDF File

You can instruct Visual CUT to add an image with an optional hotspot (a link to a web page, email, or a local file. This can be useful when you wish to use a button image as a hotspot indication or when you wish to add a company logo to a range of pages.

The command line argument structure is as follows:

…**"PDF_AddImage:c:\test.pdf>1>99>40>60>50>10>http://www.IBM.com>c:\temp\IBM.png>0"**

The parameters (after the ":") are separated by 10 ">" and are as follows:

1. **pdf file** to which the hotspot would be added.

2. **Staring with Page**

3. **Ending with Page**: use a very large number if you want to apply the hotspot to all pages and you don't know how many pages are in the document).

4. **Left** (in millimeters): The left edge of the hotspot rectangle (relative to top left corner of the page).

5. **Top** (in millimeters). The top edge of the hotspot rectangle (relative to top left corner of the page).

6. **Width** (in millimeters). The width of the hotspot rectangle (image will be resized to fit)

7. **Height** (in millimeters). The height of the hotspot rectangle (image will be resized to fit)

8. **Link.** for example: **"http://www.MilletSoftware.com" or "mailto:ido@MilletSoftware.com"** or **"file:///c:\\temp\\helloworld.bat" or "c:/temp/my_media_file.wmf"**
Leave blank if you don't want the image to act as a hotspot.

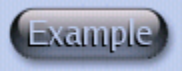
9. **Image File**: the path and name of the image file. You can use images of the following types: BMP, TIFF, JPEG, PNG, GIF, WMF and EMF.

10. **Border**: 1 for visible border, 0 for no border (default for missing or invalid value is 1)

Notes:
You can have multiple PDF_LinkToWeb command line arguments in a single command line.

You can use field or formula names within the command line argument. The dynamic content of these fields/formulas would be substituted into the argument. For example (all in one line):
…**"PDF_AddImage:{@pdf_file}>1>99>40>60>50>10>{@Link}>{@Logo}>1"**

You can use this site to create .png button images with text of your choice:
http://nyphp.org/content/presentations/GDintro/gd26.php



Adding Links/Images to Files/Pages Using Crystal Formulas as Tags

Using a command line argument, you can instruct Visual CUT to look for invisible tags inside the pdf file (Crystal formulas with font color set to white) and to replace them with links to other files. Here's an example of the command line argument structure:

… **"PDF_Link_Tags:c:\temp\Sales in {@Year_Parameter}.pdf"**
Or
… **"PDF_Link_Tags:Source_pdf_file>target_pdf_file"**

If only one pdf file is specified (as in the top example), then the source file becomes also the target file. The Crystal formulas act as tags for controlling the location, size, border, image, and behavior of the desired links.

Setting Up a Crystal Report with pdf field tags

You can download a sample report demonstrating the technique from:
www.milletsoftware.com/Download/Visual_CUT_9_PDF_Link_Tag.rpt
Sample pdf output is at: **www.milletsoftware.com/Download/Link_Test.pdf**

The sample report uses a {@PDF_Link_Tag_Sample} formula to demonstrate a **File link** to an image file and a {@PDF_Link_To_Page_N_Plus_1} formula to demonstrate a **Page link** to another page in the same document. There are no restrictions on the number or names of the formulas you can use.

The location of the link tag formula on the report layout controls the location of the file links it would generate. The links would be created for each rendered instance of the formula. For example, if you place the formula in a Group Footer you will generate a link for each Group.

The location (**top left corner**) of the rendered formula instances controls the location of the hyperlink in the pdf file. The width and height of the hyperlink hotspot is controlled by the formula text, not by the formula field size.

The formula text must be rendered all within the formula boundaries in a single line. Use very small font sizes: 2 or even 1 to achieve this. Then, turn the font color to White to make the formula invisible or, if you keep the **PDF_Tags_Delete_Default** option in DataLink_Viewer.ini as True, Visual CUT removes the tag text after processing it.

The following page demonstrates (and comments on) the structure of the required formula text.

Notes:
As always, you can use field or formula names within the command line argument. The dynamic content of these fields/formulas would be substituted into the argument.

The hyperlink hotspot rectangle created by Visual CUT doesn't provide any text. You can provide the text within the Crystal report or you can include a directive to fit an image into the hotspot rectangle. This can provide an intuitive button for the user to click & follow the hyperlink. You can use the following site to create .png button images with text of your choice:
http://www.lucazappa.com/brilliantMaker/buttonImage.php

In Crystal, **use non-proportional font** for the tag formula (avoid **Calibri** and **Bold/Italic**).
Avoid Paragraph spacing option of "Exact".

Visual CUT can remove pdf processing tags from the pdf file after processing those tags. This is controlled by an entry called **PDF_Tags_Delete_Default** under the [Options] section of DataLink_Viewer.ini. By default, this option is set to True.

Formula Example from the Sample Report

Here is the {@PDF_Link_Tag_Sample} formula from the sample report at:
www.milletsoftware.com/Download/Visual_CUT_9_PDF_Link_Tag.rpt

```
// Make the font size very small (2 or even 1 point) so that the whole
   // content displays within the formula boundaries in a SINGLE line.
// The location (top left corner) of the rendered formula instances
   // control the location of the link.
// link to pdf file in same folder: "file name"
// link to non-pdf file in same folder: ".file name"
// link to non-pdf file in 1-level higher folder: "..file name"

// The formula always starts with "#Link_Tag::" and ends with "#"
"#Link_Tag::" +
"PDF" +  "||" +  // Link Type: "PDF" for links to a page in another pdf file.
                         // "Page" for links to a page in same pdf file
                         // "File" for links to other file types.
                         // "MAIL" for mailto.
                         // "HTTP" for web link.
                            // ""  (blank) for other types of links (e.g. Tel:)
{Product.Product Name} + ".pdf" + "||" + // The file to link to. Blank if Link Type = "Page"
"" + "||" + // [optional] file to test it exists, and abort otherwise.
"1" + "||" +   // page destination in target file (ignored if not pdf).  Zoom is inherited.
                // to specify both PageN & Zoom,
                // use a ";" separator: "1;100" (page 1, zoom 100%)
                // -1 zoom value = Fit Page
                // -2 zoom value = Fit Width
"70;20" + "||" +   // hot spot boundaries in points: width;height
                         // (note: if 4 arguments are provided,
                         // they are treated as: shift_x;shift_y;width;height)
"1" + "||" +   // New Window Option. 1: opens target pdf file in a new Window
                                        // 0: opens target in current window)
"0" + "||" + // Border Option: 1: Show Border.  0: No border
"c:\temp\Click_to_View.png" +  // [optional] path & name of Image File to fit
                         // You can use images of the following types:
                         // BMP, TIFF, JPEG, PNG, GIF, WMF and EMF.
"#"
```

Embedding Files as Attachments inside a PDF File

Note: in most cases, you should skip this page. The following section describes a more powerful technique that not only embeds files, but also provides links to them.

Using a command line argument, you can instruct Visual CUT to embed files as internal attachments inside a given pdf file. The command line argument structure is as follows:

… **"PDF_Embed:PDF_File<<file1::Mime Type::title||file2::Mime Type::title…"**

The parameters (after the ":") begin with
1. **PDF_File:** The PDF file within which you wish to embed files.

Then, after a "<<" separator, you can specify any number of embedded files separated by "||"
Each embedded file is specified as 3 arguments separated by "::"

1. **File Path and Name**
   Note: If a path is not specified, the path from a previous embedded file is used

2. The **MIME Type** of the file. For example, **application/pdf** or **image/jpg**, etc.
   see http://www.w3schools.com/media/media_mimeref.asp for a list of MIME types
   If you leave blank, Visual CUT would detect the MIME type based on file extension.

3. The **Title** to **uniquely identify** the embedded file.

For example:
"PDF_Embed:**c:\temp\test.pdf**<<c:\temp\INV.pdf::application/pdf::Invoice||Sig.jpg::image/jpg::Signature"

Important Note: you can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT). The dynamic content of these fields/formulas would be substituted into the command line argument.

Adding Links and Embedded Files Using Crystal Formulas as Tags

A command line argument can instruct Visual CUT to look for tags inside the pdf file (text from Crystal formulas). When a Link_Tag with EMBED directive is found, the file specified in the tag is embedded inside the pdf file and a hotspot area designated with an icon, in the location of the tag, is created to link to that embedded file. Any file can be embedded, and embedded pdf files can have their own embedded files, which supports multi-level drill-down scenarios. See video demo: www.milletsoftware.com/Download/Visual_CUT_PDF_Embed/Visual_CUT_PDF_Embed.html

This allows you to **deliver a PDF with drill-down or added content that the user can bring up by double-clicking a link.** This is like on-demand subreports, except that:
1. the content has been pre-rendered
2. the subreports can contain their own subreports
3. the embedded files can be **any file type** (pdf, excel, Word, image, audio, video ...)
4. the **container pdf can be password protected and/or digitally signed** by Visual CUT

If the embedded file is a pdf file, PDF Xchange Viewer opens it as new tab inside the main pdf. Other viewers, such as Acrobat Reader 9 open the embedded pdf in a new viewer window.

Here's an example of the command line argument structure:
… **"PDF_Link_Tags2:c:\temp\Sales in {@Year_Parameter}.pdf"**
Or
… **"PDF_Link_Tags2:Source_pdf_file>target_pdf_file"**
If only one pdf file is specified the source file becomes also the target file.

Setting Up a Crystal Report with Link_Tag Formulas

You can download a sample report demonstrating the technique from:
www.milletsoftware.com/Download/Visual_CUT_PDF_Link_Tag_File_Embed.rpt
Sample pdf output is at:
www.milletsoftware.com/Download/Visual_CUT_PDF_Embedded_Drill_Down_Sample.pdf

The sample report uses a **{@PDF_Link_Tag_1}** formula in Group Header 1 to demonstrate a file embedding link. There are no restrictions on the number or names of the formulas you can use.

The location of the link tag formula on the report layout controls the location of the hotspot and link icon it create for each rendered instance of the formula. The **top left corner** of the rendered formula instance controls the location of the hyperlink in the pdf file. The width and height of the hyperlink hotspot is controlled by the formula text, not by the formula field size.

The formula text must be rendered all within the formula boundaries in a single line. Use very small font sizes: 2 or even 1 to achieve this. Then, turn the font color to White to make the formula invisible or, if you keep the **PDF_Tags_Delete_Default** option in DataLink_Viewer.ini as True, Visual CUT removes the tag text after processing it.

The following page demonstrates (and comments on) the structure of the required formula text.

As always, you can use field or formula names within the command line argument. The dynamic content of these fields/formulas would be substituted into the argument.

Visual CUT can remove pdf processing tags from the pdf file after processing those tags. This is controlled by an entry called **PDF_Tags_Delete_Default** under the [Options] section of DataLink_Viewer.ini. By default, this option is set to True.

Formula Example from the Sample Report

Here is the {@PDF_Link_Tag_Sample} formula from the sample report at:
www.milletsoftware.com/Download/Visual_CUT_PDF_Link_Tag_File_Embed.rpt

```
// Make the font size very small (2 or even 1 point) so that the whole
   // content displays within the formula boundaries in a SINGLE line.
// The location (top left corner) of the rendered formula controls the location of the link.
// The formula always starts with "#Link_Tag::" and ends with "#"
"#Link_Tag::" +
"Embed" +  "||" +  // Link Type. Use lower case "embed" to ignore (skip) missing files
"c:\temp\Sales for " + {Product_Type.Product Type Name} + ".pdf" + "||" + // The file to embed
"" + "||" + // MIME Type of the file to embed.  Leave blank for automatic identification
"Double-Click to See" + "||" +          // Hotspot Tooltip Header Line
"Detail for " + {Product_Type.Product Type Name} + "||" + // link description (tooltip 2nd line)
"18;18" + "||" +   // hot spot boundaries in points: width;height
                   // (4 arguments instead of 2 are treated as: shift_x; shift_y; width; height)
"0" + "||" +      // Icon Option for the link:
                  //        0 = Standard Icon (push-pin)
                  //        1 = 28x28 disk image
                  //        2 = No icon
                  //        3 = Graph
                  //        4 = Paperclip
                  //        5 = Tag
                  //        6 = Solid white rectangle
                  //        You may add 1000 to 1100 for transparency of 0% to 100%
                  //          the last digit indicates the icon. 1074 = 70% transparent Paperclip
"" +              // [optional] Image File to fit as background e.g. c:\temp\file.png
                      //supported types: BMP, TIFF, JPEG, PNG, GIF, WMF and EMF.
"#"
```

The sample PDF:
www.milletsoftware.com/Download/Visual_CUT_PDF_Embedded_Drill_Down_Sample.pdf
demonstrates a case of embedded pdf files within embedded pdf files. It was created using a batch file with 3 command lines:
1st line bursts a report to generate the lowest-level pdf files (one file for each Product).
2nd line bursts a report to Product Type PDFs and embed in them the Product PDFs.
3rd line exports the Master PDF and embeds in it the Product Type PDFs.

```
"C:\Program Files\Visual CUT 11\Visual CUT.exe" -e "C:\temp\DrillDowns2_By_Product.rpt"

"C:\Program Files\Visual CUT 11\Visual CUT.exe" -e "C:\temp\DrillDowns1.rpt"
        "PDF_Link_Tags2:c:\temp\Sales for {Product_Type.Product Type Name}.pdf"

"C:\Program Files\Visual CUT 11\Visual CUT.exe" -e "C:\temp\Master.rpt"
        "PDF_Link_Tags2:c:\Temp\Visual_CUT_PDF_Embedded_Drill_Down_Sample.pdf"
```

Note: the **PDF_Compress** command line argument can reduce the resulting pdf file size.

Adding Links to Files by Detecting File References in PDF Text

Using a command line argument, you can instruct Visual CUT to look for file references in the text of a pdf file and automatically add colored hotspots over these file references that link to the specified files. Here's an example of the command line argument structure (all in 1 line):

… **"PDF_Auto_File_Link:c:\In.pdf>>.>>.pdf;.mp3>> (>>0>>)>>255;255;0>>True>>c:\Out.pdf"**

The parameters (after the ":") are separated by ">>" and are as follows:

1. **Source pdf file** to process

2. **Base Path** where the linked files are found. There are several options here:
   a) no base path (leave **blank**) if the file references in the pdf are specified with full paths
   b) simple base path (e.g., **c:\my archive**) to prefix any file reference
   c) relative path: **.** (for same folder as the pdf file) **..** (for parent folder), etc.

3. **File Extensions** to process (separated by a semi-colon).

4. **Start Text** indicating start of file reference. In the example above, file references are assumed to always start after a **(**

5. **Number of characters to include in the file reference from the Start Text**. In the example above, no text (**0** characters) from the Start Text is included. However, there can be many cases where some text should be included. For example, if the Start Text is **(c:** then the number of characters to include should be **3**.

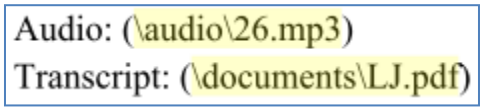
6. **End Text** indicating end of a file reference. In the example above, file references are assumed to always end with a **)** (note: can be left blank)

7. **Color of Hot Spot in RGB Values** (e.g., yellow: **255;255;0**). Note: drawn with 80% transparency, so use strong colors. (see RGB #s at: http://web.njit.edu/~kevin/rgb.txt.html)

8. True/False: **Should Visual CUT check for file existence of the file references** and generate a failure message listing all missing files (note: the output file with the generated links will still be generated, but the process will report a failure and further actions such as emailing would be aborted).

9. The Output PDF File. If left blank, the source pdf file would be updated.

Notes:
- Machine must have at least one printer installed
- Here is an example of yellow hotspots and links generated for file references detected inside a pdf file:



Detecting Additional File References Using Wild Card Tokens

The approach above relies on start and end strings. However, there are scenarios where the text to be detected doesn't have consistent start and end strings. Instead, the target text may have a predictable structure such as DOC##### where the text always starts with 'DOC' and continues with 5 digits (e.g., 'DOC01207').

When a command line already has a **PDF_Auto_File_Link** argument, you can direct Visual CUT to search for and generate links for additional targets by adding a command line argument such as this:

..."**PDF_Auto_File_Link_Tokens:**..\Multimedia**MM#####**.wmv||..\Documents**DOC#####**.pdf"

Alternatively, you can set this as a global option by adding the following entry in DataLink_Viewer.ini (under the [Options] section):

PDF_Auto_File_Link_Tokens=..\Multimedia**MM#####**.wmv||..\Documents**DOC#####**.pdf

Additional target are separated by '||' and include the relative or absolute path to the target files.

The process attempts to locate text targets with the specified file extension as well as without it. In the examples above, the first additional target is any text containing a pattern matching:
a) MM#####.wmv (for example **MM12345.wmv**)
Or,
b) MM##### (for example **MM12345**)

Note: the only wild cards allowed for this option are:

1. **#** : matched any single digit in that position
2. **?** : matches any single character (including digits) in that position

Adding Digital Signature to a PDF File

You can instruct Visual CUT to add a digital signature to a PDF file. The command line argument structure is as follows (as always, it must be all in one line):

…**"PDF_Sign:Input_File>>Output_File>>Open_Password>>Sig_Field**
… >>pfx_File>>pfx_Password>>Reason>>Location>>Contact"

The parameters (after the ":") are separated by ">>" and are as follows:
1. **Input File**: path and name of the pdf file to sign
2. **Output File**: leave blank if you wish to overwrite the original pdf file
3. **Open_Password:** if the pdf file is not password protected, leave this option blank
4. **Signature Field Name:** if the field doesn't already exist, Visual CUT creates it
5. **pfx File:** path and name of the **PKCS#12** certificate/private key file (.pfx file)
6. **pfx_Password**: password to open the pfx file
7. **Reason**: optional text indicating reason for signing
8. **Location**: optional text indicating where the document was signed (e.g., New York)
9. **Contact**: optional text indicating contact information for the signer

As always, you can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. The dynamic content of these fields/formulas would be substituted into the command line argument.

Encrypting & Protecting a PDF File

You can instruct Visual CUT to protect (using advanced 128-bit AES encryption) the exported (or any other) PDF file. The command line argument structure is as follows:

… **"PDF_PROTECT:Owner_Pass>User_Pass>1>1>0>1>1>pdf_file_path_and_name"**

The parameters (after the ":") are separated by a ">" and are as follows:

1. **Owner_Password:** Keep this password to yourself. It provides full control over the PDF file.

2. **User_Password:** Give this password to the recipient. The following arguments control what the user can do with the PDF file.
   NOTE: If you wish to protect the file but not prompt the user for a Password, leave the User_Password option blank.
   For example: **"PDF_PROTECT:Owner_Pass>>1>1>0>1>1>"**

3. **Allow User to Print the File:** (1=Yes, 0=No) **1** is typical.
4. **Allow User to Copy Text & Images from the File:** (1=Yes, 0=No) **1** is typical.
5. **Allow User to Edit/Change the File:** (1=Yes, 0=No) **0** is typical.
6. **Allow User to Add Notes to the File:** (1=Yes, 0=No) **1** is typical.
7. **Allow User to Print in Full Resolution:** (1=Yes, 0=No) **1** is typical.
   Setting this to zero would force low-resolution printing, preventing the document from being distilled into a new unrestricted PDF document.
8. OPTIONAL: The PDF file path & name (for example, **c:\temp\other_file.pdf**).
   Leaving this argument blank, would default to processing the file being exported.
   Providing a file name would direct processing to the specified file (even if it's not the exported PDF file.
9. OPTIONAL: Allow user to Fill Form Fields (if blank, assigned same value as #5 above)
10. OPTIONAL: Allow Copy for Accessibility (if blank, assigned same value as #4 above)
11. OPTIONAL: Allow user to assemble document (if blank, assigned same value as #5 above)
12. OPTIONAL: New PDF file path & name (for example, **c:\temp\protected.pdf**)
    When left blank, the protected version is saved under the original file path & name.
    Specify a new file here if you wish to retain the original unprotected version.

-- option below is new in 6.4002 ---

13. OPTIONAL: Encryption Strength: **128** = Default 128-bit AES (requires Acrobat 7 or later)
    256 = 256-bit AES (requires Acrobat 9 or later)

for example:

```
"PDF_PROTECT:123>{@Pass}>1>1>0>0>1>c:\test.pdf>1>1>0>c:\protected.pdf>256"
```

Note: you can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. The dynamic content of these fields/formulas would be substituted into the command line argument. Among other things, this allows you to **easily protect individual PDF exports with different passwords for each group.** For example (all in one line):

…"PDF_PROTECT:my_pass>{@Emp_Pass}>1>1>0>1>1>c:\{@Year} Archive\Sales For {Emp_Name} in {@Month_Name}.pdf"

Merging PDF Files

Using a command line argument, you can instruct Visual CUT to merge any number of PDF files. While each Crystal report is limited to a single paper orientation (either portrait or landscape), **this feature allows you to combine multiple report outputs even if they have different paper orientations**. The command line argument structure is as follows:

… **"PDF_MERGE:PDF_File_List>PDF_File_Target"**

The parameters (after the ":") are separated by a ">" and are as follows:

1. **PDF_File_List:** comma separated list of the source files in the order they should be merged. If all source files share the same folder, **you may specify the full path just for the first file**. As discussed in the next few pages, you can use **wild cards**, **file lists in text files**, and **dynamic references** to specify the files to be merged.

2. **PDF_File_Target:** the file path & name for the resulting merged PDF file.

3. **Optional**: **BM_By_FileName** add a bookmark for each file according to its file name.

4. **Optional**: leave blank or specify: **M1** , **M2**, or **M3**
   These codes direct Visual CUT to use alternative merge methods that differ in their speed and approach. If the default approach is not 100% successful, try one of these alternatives.

5. Optional: owner password in case one or more of the source files is password protected. All the encryption settings (owner and user passwords, and protection settings) of the last protected file in the list of source files would be applied to the resulting merged document.

Notes:

- If a source file is not found, a warning is written to Failure.log and the process continues after skipping that file.
- specifying the 4th / 5th optional argument requires that you also specify the 3rd / 4th arguments (you can leave the 3rd /4th arguments blank as in: **file list>target>>M2>>sesame**). For example:
  "PDF_MERGE:c:\temp\F1.pdf,F2.pdf,F3.pdf>c:\temp\Result.pdf>>M2>sesame"
- **M2** is fast but may have problems with input files that have different page orientations.
- **M3** should not be used when using the techniques described in "Specifying Bookmarks when Merging PDF Files" or in "Using the Merged File Names to Generate Bookmarks". It is useful in cases where M2 fails to handle certain malformed pdf files.

Dynamic File Names

You can use field or formula names within the command line argument. The dynamic content of these fields/formulas would be substituted into the command line argument.

For example, you can use Visual CUT to:
1. burst **Sales** information (**Portrait**) by Product Type into individually named pdf files
2. burst **Returns** information (**Landscape**) by Product Type into individually named pdf files
3. burst **Complaints** information (**Portrait**) by Product Type into individually named pdf files
   AND at that stage, use a command line argument to merge the 3 outputs for each Product Type, by using the following command line argument
   (Note: you should have **no line breaks** in the actual command line):

"PDF_MERGE:C:\temp\Sales_{@Prod_Type}.pdf,Returns_{@Prod_Type}.pdf,
Complaints_{@Prod_Type}.pdf>c:\temp\{@Prod_Type}.pdf"

If the current bursting cycle is for a Product_Type of "**Gloves,**" the command line argument then gets processed as:
"PDF_MERGE:C:\temp\Sales_Gloves.pdf,Returns_Gloves.pdf,Complaints_Gloves.pdf>
c:\temp\Gloves.pdf"

You can then e-mail (or staple-print) the combined PDF file to each Product Type manager.

Using a Text File to Specify Files for Merging

If you have a text file containing the list of files to be merged, such as shown below, you can instruct Visual CUT to use the text within that file as the list of files to be merged using the key word "**List_File:**" followed by the path & name of the text file. For example: "PDF_MERGE:**List_File:c:\temp\FileList.txt**>c:\temp\Result.pdf"

Using Wild Cards to Specify Files for Merging

You can also specify file names using **wild cards**.

For example, to send all pdf files in the current month folder under the current customer: "PDF_MERGE:**C:\VC_Export\{customer.cust_id}\{@Month}*.pdf**>c:\temp\Result.pdf"

and to merge all pdf files that start with the current Customer ID: "PDF_MERGE:**C:\VC_Export\{customer.cust_id}*.pdf**>c:\temp\Result.pdf"

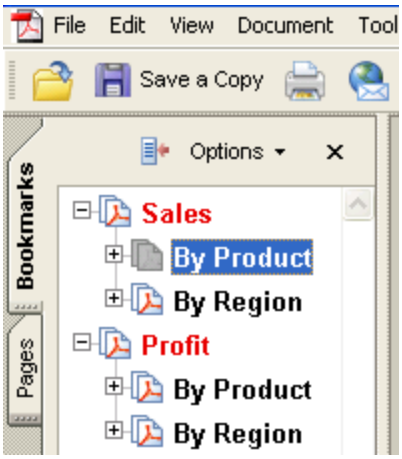
Controlling Merged Bookmark Colors

Note: these instructions are no longer needed. From November 9, 2009, Visual CUT keeps bookmark colors during merging of pdf files. Only for backward compatibility, the older functionality below is preserved.

During the merging process, bookmark colors (see Controlling PDF Bookmark Colors (& Text) on page 110) from the first file in the merged list are maintained. To apply colors to the merged bookmarks from any file, you include (anywhere) in the Bookmark Label the color constants of **crMrgRed**, **crMrgBlue** or **crMrgGreen** (note: case sensitive). Visual CUT would then set the color of the bookmark text accordingly (and remove the color constant text from the label).

Specifying Bookmarks when Merging PDF Files

During the merge process, you may want to add bookmarks for each input pdf file, so that the user can easily navigate the merged pdf file. To demonstrate how this can be done with Visual CUT, imagine you need to merge 4 pdf files:



2 pdf files with information about sales:
Product_Sales.pdf
Region_Sales.pdf

2 pdf files with information about profits:
Product_Profits.pdf
Region_Profits.pdf

You want the merged pdf to include bookmarks linking to the starting page of each of these 4 sections. The bookmarks should look like the example on the right:

Visual CUT allows you to specify, for each pdf file being merged, a chain of bookmark levels to which the content would be linked.

Let's assume you use the option to specify the list of files to merge using a text file (**c:\temp\my_list.txt**). The command line may look like this:

… "PDF_MERGE:List_File:**C:\temp\my_list.txt**>c:\temp\result.pdf"

and the **c:\temp\my_list.txt** would look like this:

c:\temp\Product_Sales{BM:Sales||+||1||225;0;0>>By Product||-||0||0;0;0}.pdf
c:\temp\Region_Sales{BM:Sales||+||1||225;0;0>>By Region||-||0||0;0;0}.pdf
c:\temp\Product_Profits{BM:Profit||+||1||225;0;0>>By Product||-||0||0;0;0}.pdf
c:\temp\Region_Profits{BM:Profit||+||1||225;0;0>>By Region||-||0||0;0;0}.pdf

The information about the top-level bookmarks associated with each file is embedded within an optional **{BM:** … **}** tag, just before the ".pdf" ending the file name.

Each node in the top-level bookmark chain is separated by **>>** and contains 4 elements separated by || (2 vertical bars):
1. Title**:** the text to appear as the label of the bookmark
2. Expand Status**:** + to open the bookmark or – to collapse it.
3. Style: 0=regular 1=Bold 2=Italic 3=Italic Bold
4. Color as RGB: 3 numbers between 0 and 255 separated by ";"
   Black=0;0;0 Maroon=225;0;0 etc. (see RGB #s at: http://web.njit.edu/~kevin/rgb.txt.html)

So the first line in the file merges Product_Sales.pdf and link its starting page to a "Sales" bookmark (expanded, bold, Maroon) and, below that, a "By Product" bookmark (collapsed, regular, Black). The second file uses the same top node and adds a "By Region" Bookmark.

Notes:

1. If the top nodes in the chain already exist, only the lower new nodes would be created. This is why in the example above, the Sales and Profit nodes are each created only once.
2. If the merged pdf file contains bookmarks, they will be transferred under the lowest node in the specified node chain for the file.
3. The resulting bookmarks are detected by the PDF_TOC command line argument, so you can automatically create a table of contents based on them.
4. You can use field or formula names within the PDF_Merge command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. You can even do the same with the content of the text file specifying the files to merge. The dynamic content of these fields/formulas would be used. In the case of specifying bookmarks during the merge process, this means that the title, style, and color of these bookmarks can be controlled via the Crystal report fields, parameters, and formula logic.

Using the Merged File Names to Generate Bookmarks

Particularly when using wildcards to merge files, but also when wishing to avoid the effort of manually specifying bookmark labels and styles, you can simply instruct Visual CUT to add bookmarks based on the file name of each merged pdf file.

To enable this functionality, you add a 3rd part to the PDF_MERGE command line argument:

```
… "PDF_MERGE:c:\{@Branch}\*.pdf >c:\temp\{@Branch} Reports.pdf>BM_By_FileName"
```

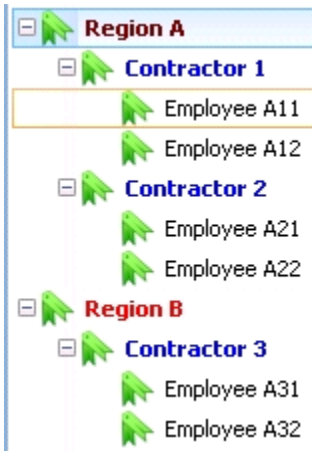
As shown in the example above, you simply add >BM_By_FileName to the end of the command line argument.

In the example above, several reports may be bursted to different {@Branch} folders. The last report may include a PDF_Merge command line argument so that each bursting step for that report will also take care of merging all the pdf files in the currently processed Branch and add bookmarks based on the file names being merged.

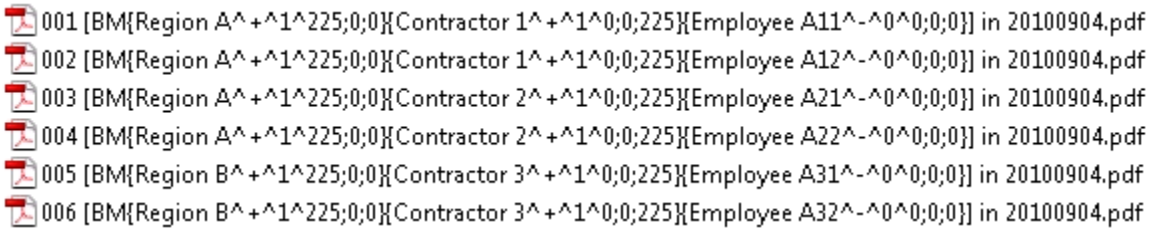
Obviously, the bookmark labels do not include the path to the pdf file nor the '.pdf' extension.

Using the Merged File Names to Generate Multi-Level Bookmarks

Imagine you need to merge 6 employee files into a PDF with 3-level bookmarks. However, you want to avoid using a text file to specify the multi-level bookmarks:



To do this, you embed the multi-level bookmark information inside the pdf file names like so:


001 [BM{Region A^+^1^225;0;0}{Contractor 1^+^1^0;0;225}{Employee A11^-^0^0;0;0}] in 20100904.pdf
002 [BM{Region A^+^1^225;0;0}{Contractor 1^+^1^0;0;225}{Employee A12^-^0^0;0;0}] in 20100904.pdf
003 [BM{Region A^+^1^225;0;0}{Contractor 2^+^1^0;0;225}{Employee A21^-^0^0;0;0}] in 20100904.pdf
004 [BM{Region A^+^1^225;0;0}{Contractor 2^+^1^0;0;225}{Employee A22^-^0^0;0;0}] in 20100904.pdf
005 [BM{Region B^+^1^225;0;0}{Contractor 3^+^1^0;0;225}{Employee A31^-^0^0;0;0}] in 20100904.pdf
006 [BM{Region B^+^1^225;0;0}{Contractor 3^+^1^0;0;225}{Employee A32^-^0^0;0;0}] in 20100904.pdf

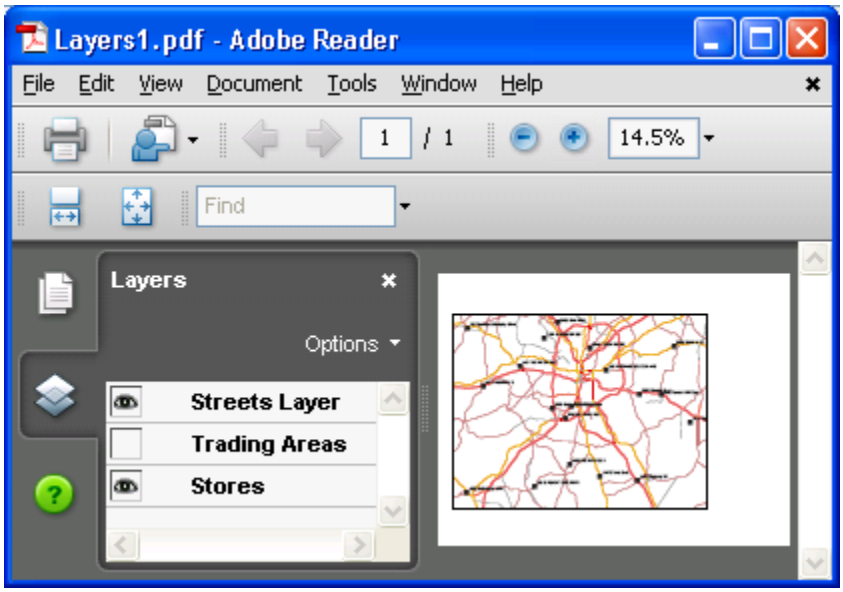
The Bookmark information is contained within the **[BM{…}]** portion. You can use the rest of the text in the file name for other purposes. In the example above, the numbers at the start of the file name are used to ensure proper alphabetical sorting when using wildcards to specify the files to be merged. For example:
... "PDF_MERGE:**c:\temp\0?? [BM{*.pdf**>c:\temp\Merge_result.pdf>BM_By_FileName"

Each Bookmark level is specified inside a **{**…**}** and has 4 elements separated by a **^** character:
1. Title**:** the text to appear as the label of the bookmark
2. Expand Status**:** **+** to open the bookmark or **–** to collapse it.
3. Style: **0**=regular **1**=Bold **2**=Italic **3**=Italic Bold
4. Color as RGB: 3 numbers between 0 and 255 separated by ";"
   Black=0;0;0 Maroon=225;0;0 etc. (see RGB #s at: http://web.njit.edu/~kevin/rgb.txt.html)

Merging 1-Page PDF Files into Layers in a Single PDF File

Using a command line argument, you can instruct Visual CUT to merge several PDF files into multiple layers inside a single pdf file. This is particularly useful for mapping applications where you users wish to turn on or off the visibility of certain map layers.

Here is an example of such a merged file, opened in Adobe Acrobat reader with the visibility of layers 1 (Streets Layer) & 3 (Stores) turned on:



The command line argument structure is as follows:

… **"PDF_MERGE_Files_to_Layers:PDF_File_List>PDF_File_Target"**

The parameters (after the ":") are separated by a ">" and are as follows:

1. **PDF_File_List:** comma separated list of the source files in the order they should be merged. If all source files share the same folder, **you can specify the full path just for the first file**. If a source file is not found, a warning is written to Failure.log and that file is skipped.

2. **PDF_File_Target:** the file path & name for the resulting merged PDF file.

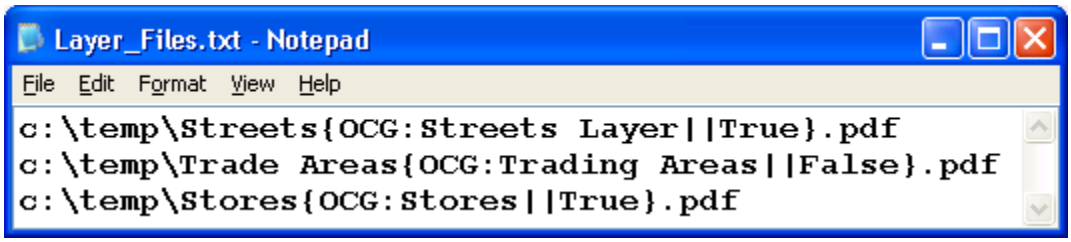
For example:
"PDF_MERGE_Files_to_Layers:c:\temp\File1.pdf,File2.pdf,File3.pdf>c:\temp\Result.pdf"

Dynamic File Names

As always, you can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT.

Using a Text File to Specify Files for Merging

If you have a text file containing the list of files to be merged, such as shown below, you can instruct Visual CUT to use the text within that file as the list of files to be merged using the key word "**List_File:**" followed by the path & name of the text file. For example: "PDF_MERGE_Files_to_Layers:**List_File:c:\temp\FileList.txt**>c:\temp\Result.pdf"



Controlling Layer Name & Visibility

By default, the name of each layer is set to its source file name (without the .pdf portion) and the visibility of each layer is turned on. You can override these default settings by embedding the desired information inside a **{OCG:** … **}** token as demonstrated by the text file sample above.

The OCG token is inserted just before the ".pdf" portion of each source file.
The token contains 2 elements separated by || (2 vertical bars):
1. Layer's name
2. Layer's initial visibility (True or False).

Printing PDF Files

Using a command line argument, you can instruct Visual CUT to print a PDF file. This is useful in combination with the previously discussed option of merging PDF files (see page 130).
It allows you to merge multiple reports, even if they use different page orientations, and then print and staple the combined output as a single print job.

The command line argument structure is as follows:

… **"PDF_PRINT:PDF_File>Printer_Name>Page_Scaling>Auto_Rotate_and_Center"**

The parameters (after the ":") are separated by a ">" and are as follows:
1. **PDF_File:** The PDF file you wish to print.

2. **Printer_Name:** the Windows name of the printer.
   Note: If you use "**Default**" as the printer name, the report gets printed to the default printer.
3. [Optional] **Page Scaling**: **None**, **Fit**, or **Shrink** (to fit paper size)
4. [Optional] **Auto Rotate** & Center: **True** or **False**

For example:
"PDF_PRINT:c:\temp\Result.pdf>\\myprintsrvr\LASER 02"
or
"PDF_PRINT:c:\temp\Result.pdf>\\myprintsrvr\LASER 02>Shrink>True"

Important Note: you can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. The dynamic content of these fields/formulas would be substituted into the command line argument.

For example, after you burst and merge multiple reports into a single PDF file for each Product Type (as described in Merging PDF Files), you can print and staple the resulting PDF file for each Product Type using the following command line argument:
"PDF_PRINT:c:\temp\{@Product_Type}.pdf>\\myprintsrvr\LASER 02>Shrink>True"

If the current bursting cycle is for a Product_Type of "**Gloves,**" the command line argument then gets processed as:
"PDF_PRINT:c:\temp\Gloves.pdf>\\myprintsrvr\LASER 02>Shrink>True"

and the combined output for just Gloves from 3 different reports (with different page orientations) would then be printed and stapled together. The process would then continue to print and staple the output for the next Product Type.

Notes:
To control number of copies printed, use the **Print_Copies** command line argument. If the dynamic value is zero, the printout would be skipped.
To control print quality/speed, use the **PDF_Print_Mode** command line argument.

Printing PDF Files To Multiple Printer Trays

You can instruct Visual CUT to split the printout of a single PDF file to multiple printer trays. For example, you may want to print the first page of a pdf file using a tray containing your company's letterhead, while the rest of the printout should use regular paper from another tray.

Visual CUT provides two ways to split the printout of a PDF file to multiple printer trays:

1. Use the **PDF_Print_Split** command line argument to specify the page ranges
2. Use the **PDF_Print_Split_Tag** command line argument to indicate that Crystal (or any other process) has embedded text tags (#Tray::tray_name/number#) in the pdf on each page or on each page where a tray change should occur.

Using PDF_Print_Split

This command line argument structure is as follows:

…

"PDF_PRINT_SPLIT:c:\temp\Test.pdf>1::\\srv2\Laser3::Top||2to5::\\srv2\Laser3::Tray 2 ||6to999::\\srv2\Laser3::Bypass Tray"

The parameters (after the ":") are as follows:

1. **PDF_File:** The PDF file you wish to print.

2. **Page_Range::Printer_Name::Tray_Name**

The 2nd element can be repeated as many times as you need, with different page ranges, different trays, and even different printers. Use "||" to separate each instance from the instance following it.

Page range can be specified as a single page or as a range, using 'to' to separate the numbers.

If you use "**Default**" as the printer name, the report gets printed to the default printer.

Trays can be specified using their names or using their internal numbers.

Important Note: you can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. The dynamic content of these fields/formulas would be substituted into the command line argument. This allows you to **dynamically control the page ranges that are sent to each printer tray!**

To control print quality/speed, use the **PDF_Print_Mode** command line argument.

Using PDF_Print_Split_Tag

The advantage of this option is its ability to dynamically control paper trays using formula logic from within Crystal reports. For example, it makes it easy to print the first and last pages in each group using one tray and print the rest of the pages from another tray.

This command line argument structure is as follows:
… **"PDF_PRINT_SPLIT_TAG:c:\temp\Test.pdf>\\srv2\Laser3"**

The parameters (after the ":") are as follows:
1. **PDF_File:** The PDF file you wish to print.

2. **Printer_Name** (use "**Default**" to print to the default printer)

Visual CUT searches each page in the pdf document for text tags that look like:
#Tray::2# (in a case where the tray number is used) or
#Tray::Bypass Tray# (in a case where the tray name is used)

Typically, these tags would be placed as Crystal formulas on the report being exported to pdf. For example, this formula (placed in the page header of the report) would cause the 1st page to use tray 3 and the rest of the pages would alternate between Tray 2 and Tray 1:

```
WhilePrintingRecords;
Global StringVar gs_Printer_Tray;

IF gs_Printer_Tray = "" THEN gs_Printer_Tray := "3"
ELSE gs_Printer_Tray := Cstr(Remainder(PageNumber,2) + 1,0);

"#Tray::" & gs_Printer_Tray & "#";
```

Notes:

1. set the font color of the formula to white, the tag would be invisible on the report as well as on the pdf file, but Visual CUT would still have access to it. Alternatively, if you keep the **PDF_Tags_Delete_Default** option in DataLink_Viewer.ini as True, Visual CUT removes the tag text after processing it. **Use non-proportional font** for the tag formula (avoid **Calibri** and **Bold/Italic**).

2. The tag doesn't have to appear on every page of the report. **Pages that don't have the tag would be printed to the paper tray last specified in previous pages**.

3. Tray numbers are not always obvious; "Tray 2" could actually be Tray #3 internally. Hence, you should probably **use Tray Names instead of Tray Numbers** in the Tags.

4. Visual CUT sets the **print job names shown in the printer queue** to reflect the page range and tray name. This makes it easy to monitor the process.

5. Make sure the printer is not setup to automatically select the paper source.

6. To control print quality/speed, use the **PDF_Print_Mode** command line argument.

Controlling Print Quality/Speed

You can use the PDF_Print_Mode command line argument to control print quality and speed. The command line argument structure is as follows:

… "**PDF_Print_Mode:Quality_Option**"

The possible Quality_Option values (after the ":") are
1 for **Normal** quality
2 for **high** quality
3 for **highest** quality

Adding Back-Pages

In some scenarios, you may need to add to each page in the exported pdf a standard back-page. For example, invoices may require some legal language on the back of each page.
Assuming you have the standard back-page as a single-page pdf file, Visual CUT can automate the process of:

a) Exporting the report to a pdf file
b) Using the **PDF_Insert_BackPage** command line argument, Inserting the standard back-page after each page and saving to a new or to the original pdf export
c) Printing the resulting pdf file via **PDF_Print** command line argument.

Since the PDF_Print command line argument is already described in the sections above, this section explains how the PDF_Inseret_BackPage command line argument is used:

Using PDF_Insert_BackPage

This command line argument structure is as follows:

… **"PDF_Insert_BackPage:c:\temp\Export.pdf||c:\temp\BackPage.pdf||c:\temp\Result.pdf"**

The parameters (after the ":") are as follows:

1. **PDF_File:** The PDF file that needs a back-page added to each page.
2. **BackPage File:** The 1-page PDF file to be inserted into the first file.
3. **The Final File**: (that's the file you would use with PDF_Print)

Notes:

1. If a Final File option is not specified, the process will simply update the source PDF_File. For example:
   … **PDF_Insert_BackPage:c:\temp\{Invoice_N}.pdf||c:\temp\BackPage.pdf"**

2. As always (as demonstrated in the example above), you can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT.

Saving PDF Files to Image Files

Using a command line argument, you can instruct Visual CUT to take a PDF file and save it to an image file. The supported image types are: BMP, JPEG, WMF, EMF, EPS, PNG, GIF, or multi-page TIFF. Here's an example of the command line argument structure:

… **"PDF_Save_As:c:\temp\Invoice.pdf>c:\temp\Invoice.bmp>BMP>72"**

The parameters (after the "**PDF_Save_As:**") are separated by a ">" and are as follows:

1. The **path & name of the PDF file** (typically, this would be the exported file, but you can convert any PDF file).

2. The **path & name of the target image file**

   Multi-Page Note: for **TIF** and **G4 TIF** export types, a multi-page pdf file results in a Multi-Page TIFF file. For all other image types, an image file for each page would be created with the page number at the end of the file name. For example, if the Invoice.pdf file in the example above had 3 pages, Visual CUT would create 3 bitmap files: Invoice**1**.bmp, Invoice**2**.bmp, and Invoice**3**.bmp

3. **Image Type**: BMP, JPEG, WMF, EMF, EMF+, EPS, PNG, GIF, HTML5, **TIF, G4 TIF**

4. **Image Quality** in Dots Per Inch (DPI): a value of 72 will give the same result as Acrobat when the zoom level is 100%.

Notes:
TIF and **G4 TIF** are not supported on Windows XP or earlier versions.

as usual, any of these arguments can contain references to fields or formulas and Visual CUT would dynamically replace the reference with the value in the report. For example:
… **"PDF_Save_As:{@export_File_Name}.pdf>{@export_File_Name}.bmp>BMP>72"**

Adding Form Fields & Submit Buttons to PDF Files

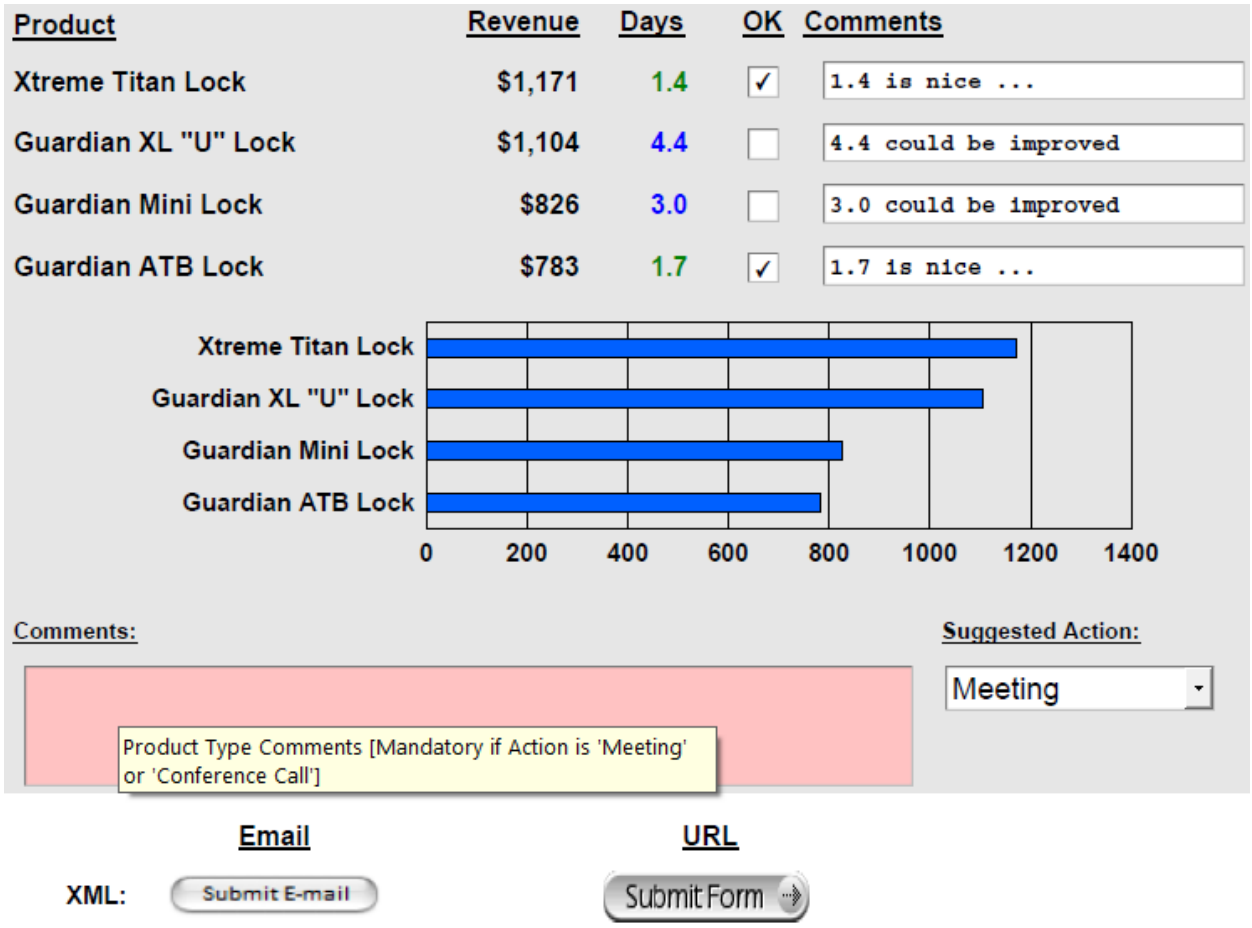
Using a command line argument, you can instruct Visual CUT to look for invisible tags inside the pdf file (Crystal formulas with font color set to background color) and to replace them with form fields such as Text, Checkboxes, Drop-Downs, and even Submit Buttons (to send form data as XML via email or URL). This means that **you can use Crystal Reports to design pdf forms, and Visual CUT to generate and distribute these forms**.

Here's an example of the command line argument structure:
… **"PDF_Form_Tags:c:\temp\Sales in {@Year_Parameter}.pdf"**

The Crystal formulas act as tags for controlling the location, size, and content of these form fields. So Visual CUT exports the report to pdf and automatically adds the form fields and any required JavaScript code. Users can then enter information into the pdf file, submit (via email or url), print, or save the completed form.

Sample PDF File with Form Fields & Submit Buttons

You can download a sample of such a pdf export with form fields at:
www.milletsoftware.com/Download/Visual_CUT_PDF_Export_with_Form_Fields.pdf



Sample Crystal Report with Formulas as Form Field Tags

You can download a sample report demonstrating the technique from:
www.milletsoftware.com/Download/Visual_CUT_9_PDF_Form_Tag.rpt

The sample report uses several formulas:

PDF_Form_Field_Product_Comment (GF2) provides short **text** area for each Product
PDF_Form_Field_Product_Checkbox (GF2) provides a **checkbox** to indicate if performance is OK

PDF_Form_Field_Product_Type_Comment (GF1) provide **multi-line text** area for each Product Type.
PDF_Form_Field_Product_Type_DropDown (GF1) provides a **DropDown** to indicate required Action

PDF_Form_Field_Submit_Button_Email_XML (RF) **Button** to submit form data as XML via **email**
PDF_Form_Field_Submit_Button_URL_XML (RF) **Button** to submit form data as XML via **url**

PDF_Form_Field_Text_Secret (RH) provides a **hidden** form field (to include in XML)
PDF_Form_Field_Text_URL (RH) provides a visible but **uneditable** form field

Setting Up a Crystal Report with pdf field tags

There are no restrictions on the number or names of the formulas you can use. However, each rendered instance must provide a **unique** form field title (avoid dots). The formula text must be rendered all within the formula boundaries in a single line. Use very small font sizes: 2 or even 1 to achieve this.
Use non-proportional font for the tag formula (avoid **Calibri** and **Bold/Italic**).

Obviously, you may not want to show the formula text on the resulting pdf file. You can turn the font color to White (or same as section background) to make the formula invisible. Better yet, if you keep the **PDF_Tags_Delete_Default** option in DataLink_Viewer.ini as True, Visual CUT removes the tag text after processing it. In order to allow Visual CUT to remove the tag text after it has been processed, be sure to replace any double quotes or parentheses in the tag text with single quote or square brackets.

Each formula instance (Group Footer/Header as well as Page Header/Footer formulas may have many instances) can trigger the creation of a pdf form field. The location (**top left corner**) of each rendered formula instance controls the location of the pdf form field. The width and height of the form field is controlled internally -- not by the formula field size.

Note: to allow recipients of the pdf file to save the completed form, I recommend the following free pdf viewer: http://www.tracker-software.com/product/pdf-xchange-viewer

Formula Example from the Sample Report

The following page demonstrates and comments the structure of the required formula text for one of the formulas in the sample report. You can open the sample report to see the fully-commented syntax for the other form field types..

Here is the {@PDF_Form_Field} formula from the sample report at:
www.milletsoftware.com/Download/Visual_CUT_9_PDF_Form_Tag.rpt

```
// Make the font size small (even 1 point). Text must stay in formula boundaries in SINGLE line!
// Turn the font color to White to make the formula invisible. Or, if PDF_Tags_Delete_Default in
// DataLink_Viewer.ini is True, tag text is removed automatically after processing.
// Location (top left corner) of the rendered formula instance control location of pdf form field.

// The formula always starts with "#Form_Tag::" and ends with "#"
"#Form_Tag::" +
"350;50" + "||" +   // field boundaries in points: width;height (or shift_x;shift_y;width;height)
{Product_Type.Product Type Name} + "_Comments" + "||" + // UNIQUE title of form field
"TEXT" + "||" + // type of form field: TEXT/DropDown/CheckBox/Submit_Button
"0" + "||" + // max number of allowed characters.  Set to "0" if no restriction.
"NotComb" + "||" + // "Comb" or "NotComb" field (if Comb, specify max number of characters
                   // above to create an equally spaced "comb" field effect)
"Left" + "||" +          // alignment:  "Left", "Center" or "Right"
"1;Inset;0;0" + "||" +   // Field Border: line width;style (Solid, Dashed, Beveled, or Inset);
                         // length of dash;length of dash space
"230;230;230" + "||" +          // Border color using RGB (0;0;128 = Navy Blue)
"255;255;255" + "||" + // Background Color ("255;248;220" = cornsilk1, "255;255;255" = White)
"0;0;0" + "||" +         // Text Color ((0;0;0 = Black)
// Possible font types: Courier, CourierBold, CourierBoldItalic, CourierItalic
                        // Helvetica, HelveticaBold, HelveticaBoldItalic, HelveticaItalic
                        // TimesRoman, TimesBold, TimesItalic, TimesBoldItalic
"CourierBold;10" + "||" + // Font type and size
// Initial Field Text Value (Use "" to specify an empty field)
iif({Product_Type.Product Type Name} in ["Saddles", "Locks"], "", "Comments on the
information for " + {Product_Type.Product Type Name} + ": ") +  "||" +
"NoScroll" + "||" +     // Scroll or NoScroll
"MultiLine" + "||" +    // SingleLine or MultiLine
"SpellCheck" + "||" +  // SpellCheck or NoSpellCheck
// --- new options in 6.2002: ---
// tool tip shown when hovering over field. Use "Same_As_Text" to display Field Text value
"Product Type Comments [Mandatory if Action is 'Meeting' or 'Conference Call']" + "||" +
"False" + "||" + // Read Only status. If True, user can see but not change the value
"False" + // Hide Field? Useful when user should not see a submitted value
"#"        // The formula always ends with "#"
```

- VC removes pdf processing tags from the pdf file (after processing) if **PDF_Tags_Delete_Default** under the [Options] section of DataLink_Viewer.ini is left as True. To ensure all tags are removed, use Replace() to change double quotes into single. For example, the form field title can be:
  Replace({Product_Type.Product Type Name},"""","'") + "_Comments"
- To accommodate more text in the tag, you may embed references to fields/formulas in it.
- To specify child form fields, specify a title of Parent_Field_title.Child_Field_Title.

Populating Form Field Text or Description via ODBC Query

When dealing with multiline text form fields, you may need to populate the text of the field with very long text that might cause the tag content to spill over more than 1 line. To address that scenario you may embed a directive to execute a SQL Query via ODBC inside the options for ***Initial Text*** or ***Field Description*** (tooltip) and Visual CUT would replace the directive with its dynamic result.

Here is an example of what the Crystal formula may look like:

```
"#Form_Tag::" + "350;50" + "||" +  {Product_Type.Product Type Name} + "_Comments" + "||"
+  "TEXT" + "||" + "0" + "||" + "NotComb" + "||" + "Left" + "||" + "1;Inset;0;0" + "||" +
"230;230;230" + "||" + "255;255;255" + "||" + "0;0;0" + "||" + "CourierBold;10" + "||" +
// Initial Field Text Value (populated via SQL Query
"{ODBC:Customers:: :: ::SELECT [Name] FROM [CUST] WHERE [Cust_ID] = 6}" +  "||" +
"Scroll" + "||" + "MultiLine" + "||" + "SpellCheck" + "||" +
"Product Type Comments [Mandatory if Action is 'Meeting' or 'Conference Call']" + "||" +
"False" + "||" + "False" + "#"
```

The expression starts with **`{ODBC:`** followed by 4 elements separated by **`::`** and ends with **`}`**

1. **`ODBC DSN`** (Note: could be different from the DSN used for the Crystal report)
2. **`User ID`** (use a single space if not needed, as shown above)
3. **`Password`** (use a single space if not needed, as shown above)
4. **`SQL Statement`**

MS Access Example:

{ODBC: Customers::User_ID::Password::SELECT [email] FROM [Contacts] WHERE [Customer_ID] = '301'**}**

SQL Server Example:

{ODBC:CONTACTS**::**sa::xxxxx::SELECT AHD.ctct.Comments FROM AHD.ctct where AHD.ctct.Comments IS NOT NULL**}**

Notes:

- If the SQL statement returns multiple rows, Visual CUT takes care of concatenating the text from all the rows to a single string with carriage return and line feed as the delimiter.
- The SQL statement syntax depends on your database. For example, as shown in the samples above, MS Access uses [ ] around field/table names but SQL Server doesn't.
- You would typically build the expression while using some field/formula values (not names) to control the WHERE clause.

Filling PDF Forms

Using a command line argument, you can instruct Visual CUT to fill the fields in a given PDF Form and save the result into a new PDF file. Here's an example of the command line argument structure:

… **"PDF_FORM:c:\temp\MyForm.pdf>False"**

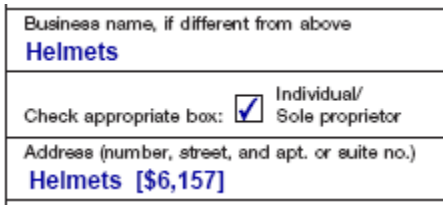
The parameters (after the "**PDF_FORM:**") are separated by a ">" and are as follows:

1. The **path & name of the PDF form file** (used as a template).

2. **Flatten** form fields (True or False)? If True, as field values get substituted by formula values from the Crystal report, Visual CUT turns the form field into a static portion of the PDF file. This means the field will no longer act as a field in the new pdf file. It will simply behave as a static portion of the new PDF file.

Setting Up a Crystal Report to Act as a PDF Form Filler

Each field in a PDF form has a name ("title" in pdf lingo). In order to fill (or substitute) the value of such a field, your Crystal report should have a string formula with the same name as the PDF form field. For example, if you desire to fill a PDF form field called "**f1-1**", you should create a Crystal formula called "**f1-1**" providing the desired value for that field. In order to supply a value for a checkbox field (as shown below), use "Yes" as the value for checked and "No" for the value for unchecked.



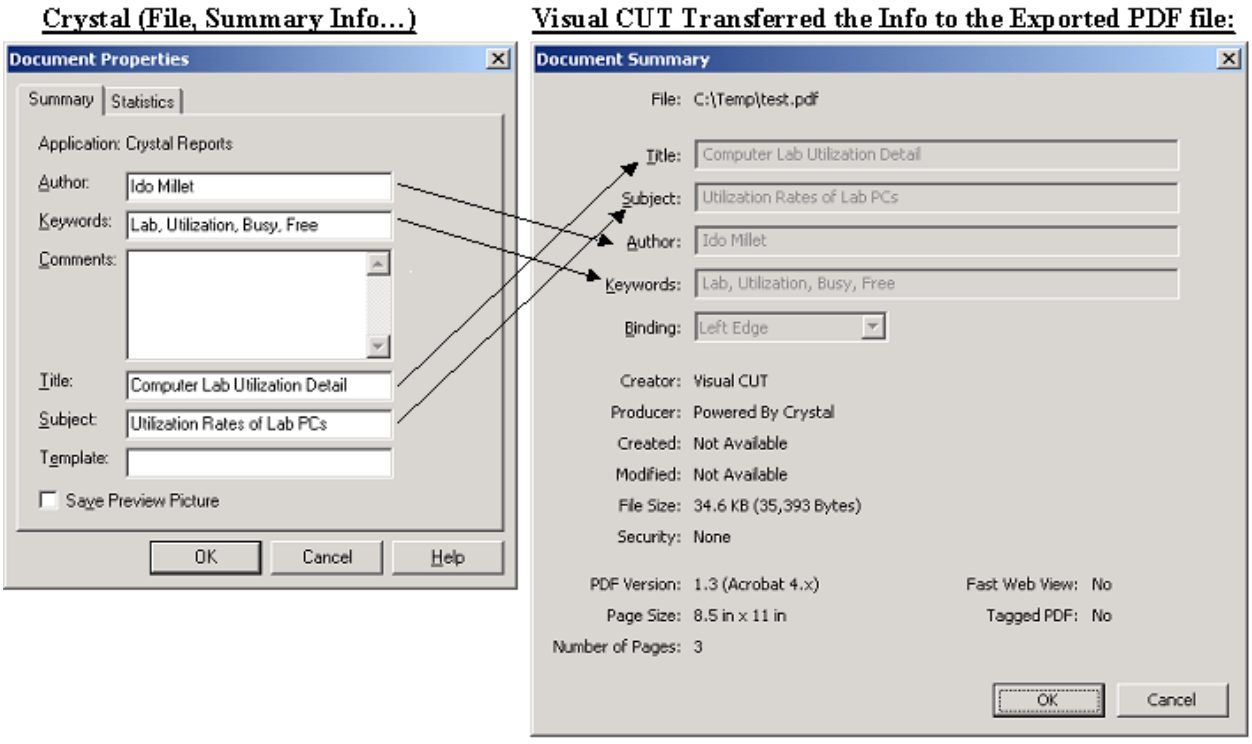
The Crystal formulas should be placed in the report header/footer if the value is the same for all records in the report. It should be placed in Group Level 1 Header/Footer if the value changes for each Level 1 group and you are bursting (generating a different PDF file for each Level 1 Group). As usual, you may suppress the formulas.

Setting Up the Report in Visual CUT

In Visual CUT, simply set the report to export to PDF and specify the export file name. As usual, you may make the export file path and/or name dynamic (reflecting field/formula values from the report. Note that if you use the PDF_FORM command line argument, Visual CUT will not export the report to PDF. Instead, Visual CUT would substitute the form fields in the Form File (specified in the command line argument) and then save the result to the specified export file name.

Setting PDF Document Properties after Export (Automatic)

After PDF exports, Visual CUT sets the PDF document summary information according to the summary information set for the report in Crystal.



This functionality, as well as the PDF Bookmarks functionality described in the previous section, are **enabled by default**. To turn it off, use the Options dialog and uncheck the option of "PDF Exports include Document Properties and Bookmark Processing."

Flatten PDF Files

You can flatten form fields and annotations in pdf files via the command line argument of **PDF_Flatten**. Here's an example of the command line argument structure:

…**"PDF_Flatten:c:\Before*.pdf>>MyPassword>>c:\After\"**
or (for a case with no password):
…**"PDF_Flatten:c:\Before*.pdf>>>>c:\After\"**
or (for a case without target folder:
…**"PDF_Flatten:c:\Before*.pdf>>MyPassword>>"**
or (for a case without password and no target folder:
…**"PDF_Flatten:c:\Before*.pdf>>>>"**

The parameters (after the "**PDF_Flatten:**") are separated by a ">>" and are as follows:

1. **PDF file(s):** comma separated list. Can use wild cards!
2. **Password** (leave blank if the pdf files are not password protected)
3. **Target Folder** (leave blank if the flattened pdf files should replace the original files)

Setting PDF Document Properties

You can specify the pdf document properties via the command line argument of **PDF_Properties**. Here's an example of the command line argument structure:

…**"PDF_Properties:c:\test.pdf>Ido>{@Invoice_N}>Invoice>Invoice>Ido>Millet Software"**

The parameters (after the "**PDF_Properties:**") are separated by a "**>**" and are as follows:

1. **PDF file(s):** comma separated list. Can use wild cards!
2. **Author of the PDF file**
3. **Title**
4. **Subject**
5. **Keywords**
6. **Creator**
7. **Producer**

   ----
8. **Page Mode:** ***Normal*** (or blank)/***Outlines/Thumbnails/FullScreen/OCG/Attachments***
9. **Open Page** (1 is default)
10. **Zoom** **(75** for 75%, **0** for default**, -1** for **fit in window**, **-2** for **fit width)**

    ----
11. **Custom Properties (Name1::Value1||Name2::Value2)**
    For example, **"…>Case Type::Abduction||Case Manager::Officer Krupke"**
    Any number of pairs delimited by **||** with a **::** separating Key Name from Key Value.

Notes:

- To target multiple files, specify a list separated by commas.
- You can use wild cards to target multiple files (e.g. **c:\temp\test.pdf,c:\temp\Inv*.pdf**)
- Specify the first 7 elements, or all 10 elements, or all 11 elements
- As usual, any of these elements can contain references to fields or formulas and Visual CUT would dynamically replace the reference with the value in the report.
- Page Mode controls what panels are visible when the pdf file is initially opened.
  Outlines: Show the bookmarks pane
  Thumbnails: Show the thumbnails pane
  FullScreen: Show the pdf in full screen mode
  OCG: shows the ***Optional content group*** pane
  Attachments: shows the attachments pane

- If you wish to avoid setting/overriding any of these properties, just leave them blank. For example, to set just the author and the subject, the command line may look like this:
  …**"PDF_Properties:c:\Inv*.pdf,c:\test.pdf>Ido>>{@Invoice_N}>>>>Outlines>1>100"**

- Specifying **Custom Properties** is the same as manually specifying them in Acrobat Pro using File, Properties, Custom Tab. These properties show up in the list of document properties and can be useful for users and for document management systems.

Building an Index PDF Documents

You can build a full-text index for multiple pdf files via the command line argument of **PDF_Build_Index**. Here's an example of the command line argument structure:

…**"PDF_Build_Index:c:\temp\dir1||c:\temp\dir2>c:\temp\Index.pdx>"**

The parameters (after the "**PDF_Build_Index:**") are separated by a "**>**" and are as follows:

1. **List of folders (separated by "||") containing pdf files to be indexed**
   Notes**:**
   a. All pdf files found in these folders and all subfolder (recursive) will be indexed.
   b. Do not include a backslash at the end of each folder path.

2. **Index (.pdx) file path and name**
   Notes:
   a. The pdx file will be deleted, if it already exists, when the process starts
   b. If an "Index" folder exists under the folder destination for the pdx file, that folder is deleted when the process starts because this folder needs to be created from scratch by the indexing process.
3. **Options:** leave blank. Reserved for future needs such as specifying maximum amount of time the process should require. The default is 2 hours.

Notes:
On the Visual CUT machine, Acrobat Pro must be installed and the build index button must be located on the toolbar like this:

As usual, any of these arguments can contain references to fields or formulas and Visual CUT would dynamically replace the reference with the value in the report. For example:
… **"PDF_Build_Index:{@Folders}>c:\temp\Index.pdx>"**

Adding an Index File to a PDF Document

You can add an index (.pdx file) reference to a pdf document via the command line argument of **PDF_Add_Index**. Here's an example of the command line argument structure:

…**"PDF_Add_Index:c:\test.pdf>c:\textMyIndex.pdx>PDX"**

The parameters (after the "**PDF_Add_Index:**") are separated by a "**>**" and are as follows:

1. **PDF file path & name**
2. **Index (.pdx) file path and name** (on the machine where the pdf file will be used)
3. Index Label **(must always be "PDX" -** Other values would be converted to "PDX"**)**

Note: Index file path can be specified as **relative** to the PDF file location.
For example: **..\Index.pdx**

Adding Multimedia to PDF Document

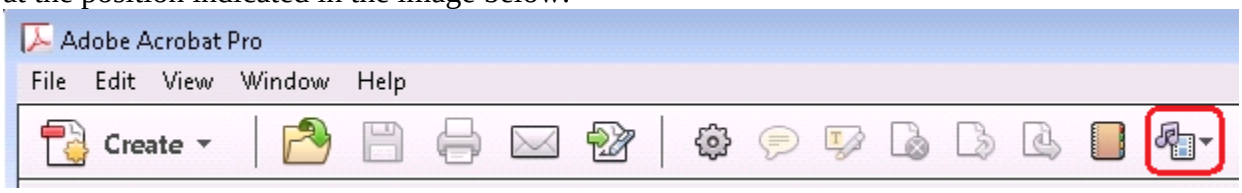
You can ask Visual CUT to add a sound file to be played when the pdf file is opened in Acrobat Reader via the command line argument of **PDF_Add_Media**. Here's an example of the command line argument structure:

…**"PDF_Add_Media:Pdf_In>>Pdf_Out>>Media_File>>1>>X>>Y>>Width>>Height>>Options"**

The parameters (after the "**PDF_Build_Index:**") are separated by a "**>>**" and are as follows:

1. **PDF File to open**
2. **PDF File to Save As** (leave blank if same as source file above)
3. **Media File** (e.g. c:\temp\Greeting.mps)
4. **Page where the multimedia box would be added** (currently, must be 1)
5. **X: horizontal offset of the multimedia box**
   Note: relative to drop-down highlighted in image below. Can be negative (e.g. -20)
6. **Y: vertical offset of the multimedia box**
7. **Width** of multimedia box
8. **Height** of multimedia box
9. **Media Type**: currently must be "Sound"
10. **Options:** leave blank. Reserved for future needs.

Notes:
Acrobat Pro must be installed and the Add Multimedia dropdown must be located on the toolbar at the position indicated in the image below:



You may need to experiment with the coorindates (X, Y, Width, Height) to avoid conflict with existing objects on the page.

As usual, any of these arguments can contain references to fields or formulas and Visual CUT would dynamically replace the reference with the value in the report.
For example (all in 1 line):

…
"PDF_Add_Media:{@Catalog}>>{@Catalog_with_Greeting}>>c:\temp\Greeting.mp3>>1>>-20>>100>>140>>16>>Sound>>"

Importing Multi-Page TIFF Files into PDF Files

Imagine you wish to use Visual CUT to burst a Crystal report and email customers invoices as PDF files. The customers wish to see documentation (scanned as multi-page TIFF files) included at the end of each invoice. Since Crystal can only display the first page of such TIFF files, you need a way to append the related TIFF file at the end of each invoice, before emailing the resulting pdf file to the customer.

Using a command line argument, you can instruct Visual CUT to take a multi-page TIFF file and turn it into a pdf file or append it to the end of an existing pdf file.
Here's an example of the command line argument structure:

…**"PDF_From_TIFF:c:\tmp\Invoice_{@Inv_N}.pdf>{@Tiff_File}>0>True>True"**

The parameters (after the "**PDF_From_TIFF:**") are separated by a "**>**" and are as follows:

1. The **path & name of the PDF file** (typically, this would be the exported file, but you can use any PDF file). In bursting scenarios, you would typically refer to the dynamic exported file name by embedding a reference to a field or formula (marked in blue in the example above).

2. The **path & name of the TIFF file(s)**
   A **semi-colon (;) separated** list of the TIFF files in the order they should be merged into the pdf file. If all source files share the same folder, you can specify the full path just for the first file. If a source file is not found, a warning is written to Failure.log and that file is skipped.
   Typically, you would use a field or a formula (marked in blue in the example above) from the report to dynamically provide the path and name of the TIFF file(s) related to the Group Level 1 in the current bursting step.

3. **Scaling relative to original image size**: **100** = original size. **120** = 20% larger.
   0 = requests automatic scaling of the image to fully occupy the page size.

4. **Append** (True/False): set to true if the TIFF should be appended to the pdf file, if that target file already exists. Otherwise, if the pdf file already exists, it would be transferred to the recycle bin and the TIFF content would be used to create a new version of that pdf file. When Append is set to False, page dimensions are controlled by the TIFF image dimensions and scaling, rather than by the page dimensions of the "receiving" pdf file.

5. **Critical** (True/False): controls what should happen if the TIFF file is not found. If set to TRUE, Visual CUT processing would be aborted. If set to False, only a Warning is logged but processing continues (if the Invoice pdf can't find a matching TIFF file, it would be emailed without that content).

Note: as usual, any of these arguments can contain references to fields or formulas and Visual CUT would dynamically replace the reference with the value in the report.

Importing Image Files as New Pages (Tag Approach)

Imagine Visual CUT burst an invoices report to PDF for emailing to each customer. Each customer may have multiple invoices and you wish to insert at the end of each invoice an image or multi-page TIFF file reflecting a scanned proof-of-delivery. Visual CUT can do this if you include on the last page for each invoice a tag specifying the image or multi-page TIFF file that should be inserted after that page in the exported pdf file.

Supported image types include: PNG, GIF, BMP, JPEG/JPG, PNG, GIF, WMF, EMF, TIF/TIFF

Here's an example of the command line argument structure:
… **"PDF_Insert_Pages_Tags:c:\temp\{@Cust}_Invoices.pdf"**
Or
… **"PDF_Insert_Pages_Tags:c:\temp\{@Cust}_Invoices.pdf>>c:\temp\{@Cust}_Invoices.pdf"**

If only one pdf file is specified (as in the top example), then the source file becomes also the target file. Otherwise, the resulting file is saved to the 2nd pdf file.

You can download a sample report demonstrating the technique from:
http://www.milletsoftware.com/Download/VC11_PDF_Insert_Pages_Tags.rpt

The formula acting as a tag can be placed anywhere on the report. It specifies what image files should be inserted as following pages. In this sample report, the tag formula is called **{@Insert_Pages_Tag}** and is located in the page footer. The formula name doesn't matter, but the logic should follow this example:

```
"#Insert_Pages_Tag::" +
"C:\temp\sample1.jpg;sample2.tif" + // list of files. Wild cards OK.
"||" +  // Separator
"100" + // Scaling: 100 = original. 120 = 20% larger, etc. 0 = auto-fit to page
"#"
```

The formula text must be rendered all within the formula boundaries in a single line. Use very small font sizes: 2 or even 1 to achieve this. You may turn the font color to White (or same color as the background) to make the formula invisible.

Notes:

1. In Crystal, **use non-proportional font** for the tag formula (avoid **Calibri** and **Bold/Italic**)

2. As always, you can use field or formula names within the command line argument. The dynamic content of these fields/formulas would be substituted into the argument.

3. In the list of files, if path is same, you may specify just the file name for subsequent files.

4. The list of files can use wild cards (e.g. **c:\scans\{cust_id}*.tiff**).

5. Visual CUT can remove pdf processing tags from the pdf file after processing those tags. This is controlled by an entry called **PDF_Tags_Delete_Default** under the [Options] section of DataLink_Viewer.ini. By default, this option is set to True.

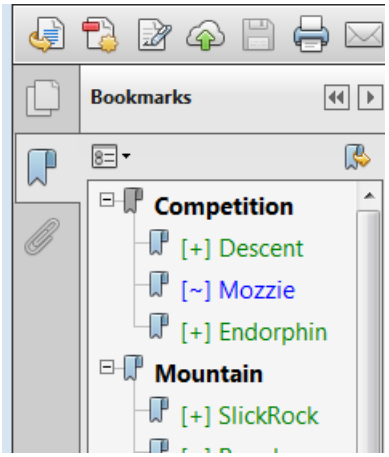
Splitting PDF Files

Visual CUT provides two ways to split pdf files:

1. **Using pdf bookmarks**, VC can create a new pdf file for each bookmark at a specified level and name the pdf file based on the bookmark labels and as well as based on Crystal report formulas.
2. **Using invisible text tags** placed on the pdf pages (typically Crystal formulas with font color to make them invisible). These tags act as split directives indicating file names for the pages starting with the tag location and ending with the next tag. These tags can also direct Visual CUT to encrypt the resulting pdf files with different passwords. Visual CUT takes care or removing these tags from the resulting pdf files.

Splitting By Bookmarks

Imagine you have a pdf file with bookmarks at 2 levels: Product Type & Product. The bookmarks were generated during a Crystal report export by Visual CUT, or perhaps they are part of an existing pdf file. Using the command line argument of **PDF_Split_By_Bookmarks** you can direct VC to Split the pdf to a separate pdf file for each bookmarks at a specified level, name the file based on the bookmark label, and even place the resulting file in a dynamically specified/created folder based on the bookmark name.



Here's an example of the command line argument structure (all in 1 line):
… **"PDF_Split_By_Bookmarks:c:\Sales.pdf>>2>>**
c:\{[Bookmark_Name]}\Sales for {[Bookmark_Name]} in {[yyyy]}.pdf

The parameters (after the "**PDF_Split_By_Bookmarks:**") are separated by a "**>>**" and are as follows:

1. The **path & name of the source PDF file**
2. **The targeted bookmark level** (level **2** in this example targets the Product level)
3. The **path & name of the split pdf file** to create for each targeted bookmark

A **{[Bookmark_Name]}** reference gets replaced by the bookmark label. For example, **Mozzie**
A **{[Bookmark_Chain_A]}** reference gets replaced by a hyphen-separated sequence of bookmark labels leading to/including the targeted node. For example: **Competition – Mozzie**

Notes:

- For each targeted bookmark, a separate pdf is created from the page associated with the bookmark up to (but not including) the page associated with the next targeted bookmark.
- If the target folder doesn't exist, VC creates it on the fly.
- Invalid file name characters are automatically substituted with valid ones.
- If a path to the target folder is not specified, the path to the source file is used.
- As usual, any of these arguments can contain references to fields or formulas, and VC would dynamically replace the reference with the value in the report. In the example above, the {[yyyy]} gets replaced with the current year.
- The process is very fast

Splitting By Embedded Tags

Using a command line argument, you can instruct Visual CUT to look for invisible tags inside the pdf file (Crystal formulas with font color set to background color) and to split the file into multiple pdf files. The following situations can benefit from this splitting of a pdf file (as opposed to bursting the Crystal report directly to multiple pdf files):

1. **Speed of bursting is incredible**, particularly for huge pdf files with many group values.
2. You can split the main pdf file based on any logic (e.g., level 4 grouping).
3. You can split any pdf file, even if it was not exported from Visual CUT.
4. You can "Print" reports to pdf before splitting (avoiding pdf export font issues)

Here's an example of the command line argument structure:

… **"PDF_Split_Tags:c:\temp\Sales.pdf>>True>>True>>False"**

The parameters (after the "**PDF_Split_Tags:**") are separated by a ">>" and are as follows:

1. The **path & name of the PDF file**
2. **Compress the split pdf files?** (set to **False** for faster execution)
3. **Create missing folders?** (set to **False** for faster execution)
4. **Assume Tag is always Nth text object on page?** (set to **True** for faster execution)

Within the pdf file, you should embed text tags (Crystal formulas) that indicate the page locations for starting a new split pdf file, and the path & name of that new file.

Notes:

- If the target folder of the split files does not exist, Visual CUT creates it on the fly.
- In Crystal, **use non-proportional font** for the tag formula (avoid **Calibri** and **Bold/Italic**)

Here is an example of such a formula that can be placed, for example, in a Group Header section:

```
// Set small font. Text must display within the boundaries in a SINGLE line.
// The page where the tag is detected becomes the first page of the split pdf
// The last page of the split pdf is the page before the next tag.
// If a single page has 2 Split Tags, only the first one is detected.

// The formula always starts with "#Split_Tag::" and ends with "#"
"#Split_Tag::" +
"c:\" + {@Folder} + "\" + {Product_Type.Product Type Name} + ".pdf" + // Split PDF File
"#"
```

Notes: Visual CUT can remove pdf processing tags from the pdf file after processing those tags. This is controlled by an entry called **PDF_Tags_Delete_Default** under the [Options] section of DataLink_Viewer.ini. By default, this option is set to True. To ensure all tags are removed, use the Replace() function in Crystal to change double quotes into single quotes. For example, the expression for the file name could be:
Replace({Product_Type.Product Type Name},"""","'")

Splitting and Protecting PDF Files By Embedded Tags

Using a command line argument, you can instruct Visual CUT to look for invisible tags inside the pdf file (Crystal formulas with font color set to background color) and to split the file into multiple **protected** pdf files. The following situations can benefit from this splitting of a pdf file (as opposed to bursting the Crystal report directly to multiple pdf files):

5. **Speed of bursting is incredible**, particularly for huge pdf files with many group values.
6. You can split the main pdf file based on any logic (e.g., level 4 grouping).
7. You can split any pdf file, even if it was not exported from Visual CUT.

Here's an example of the command line argument structure (all in 1 line):

… **"PDF_Split_Protect_Tags:c:\temp\Sales.pdf>>False>>True>>False…**
>>Owner_Password>>1>>1>>0>>1>>1>>1>>1>>0"

The parameters (after the "**PDF_Split_Protect_Tags:**") are separated by a "**>>**" and are as follows:

1. The **path & name of the PDF file**
2. **Compress the split pdf files?** (set to **False** for faster execution)
3. **Create missing folders?** (set to **False** for faster execution)
4. **Assume Tag is always Nth text object on page?** (set to **True** for faster execution)
5. **Owner_Password:** Keep this password to yourself. It provides full control over the PDF file. Note: user password is provided in the Tag embedded in the pdf.
6. **Allow User to Print the File:** (1=Yes, 0=No) **1** is typical.
7. **Allow User to Copy Text & Images from the File:** (1=Yes, 0=No) **1** is typical.
8. **Allow User to Edit/Change the File:** (1=Yes, 0=No) **0** is typical.
9. **Allow User to Add Notes to the File:** (1=Yes, 0=No) **1** is typical.
10. **Allow User to Print in Full Resolution:** (1=Yes, 0=No) **1** is typical.
    Setting this to zero would force low-resolution printing, preventing the document from being distilled into a new unrestricted PDF document.
11. **Allow user to Fill Form Fields**: (1=Yes, 0=No) **1** is typical.
12. **Allow Copy for Accessibility**: (1=Yes, 0=No) **1** is typical.
13. **Allow user to assemble document**: (1=Yes, 0=No) **0** is typical.

Within the pdf file, you should embed text tags (Crystal formulas) that indicate the page locations for starting a new split pdf file, and the path & name of that new file.

Notes:

- If the target folder of the split files does not exist, Visual CUT creates it on the fly.
- In Crystal, **use non-proportional font** for the tag formula (avoid **Calibri** and **Bold/Italic**)

Here is an example of such a formula that can be placed, for example, in a Group Header section:

```
// Set small font. Text must display within the boundaries in a SINGLE line.
// The page where the tag is detected becomes the first page of the split pdf
// The last page of the split pdf is the page before the next tag.
// If a single page has 2 Split Tags, only the first one is detected.

// The formula always starts with "#Split_Protect_Tag::" and ends with "#"
"#Split_Protect_Tag::" +
"c:\" + {@Folder} + "\" + {Product_Type.Product Type Name} + ".pdf>>User_Password"
// NOTE: to protect but not prompt the user for a Password, leave User_Password as blank. +
"#"
```

Notes: Visual CUT can remove pdf processing tags from the pdf file after processing those tags. This is controlled by an entry called **PDF_Tags_Delete_Default** under the [Options] section of DataLink_Viewer.ini. By default, this option is set to True. To ensure all tags are removed, use the Replace() function in Crystal to change double quotes into single quotes. For example, the expression for the file name could be:
Replace({Product_Type.Product Type Name},"""","'")

Compress PDF Files

Using a command line argument, you can instruct Visual CUT to compress a pdf file.
Here's an example of the command line argument structure:

…**"PDF_Compress:c:\tmp\Inv_{@N}.pdf>>c:\tmp\Inv_{@N}_New.pdf"**

The parameters (after the "**PDF_Compress:**") are separated by a "**>>**" and are as follows:

1. The **path & name of the existing PDF file** (typically, this would be the exported file, but you can use any PDF file). In bursting scenarios, you would typically refer to the dynamic exported file name by embedding a reference to a field or formula (marked in blue in the example above).

2. [optional] The **path & name of the new PDF file**
   If you leave this part blank, Visual CUT would simply overwrite the pdf file with its new compressed version. For example:
   …**"PDF_Compress:c:\tmp\Inv_{@N}.pdf"**
   Use that approach only if you have a way to recreate or restore the original file because the compress process might fail.

Notes:
As usual, any of these arguments can contain references to fields or formulas, and Visual CUT would dynamically replace the reference with the value in the report.

PDF_Compress occurs before **PDF_Protect** to allow compressing and protecting in one command line.

Linearize (web-enable) PDF Files

A linearized pdf file can be opened faster from a url link because it is designed to open the 1st page even before the full document has been downloaded by the browser.

Using a command line argument, you can instruct Visual CUT to linearize a pdf file.
Here's an example of the command line argument structure:

…**"PDF_Linearize:c:\tmp\Inv_{@N}.pdf>>c:\tmp\Inv_{@N}_New.pdf>>Password"**

The parameters (after the "**PDF_Linearize:**") are separated by a "**>>**" and are as follows:

3. The **path & name of the existing PDF file** (typically, this would be the exported file, but you can use any PDF file). In bursting scenarios, you would typically refer to the dynamic exported file name by embedding a reference to a field or formula (marked in blue in the example above).

4. [optional] The **path & name of the new PDF file**
   If you leave this part blank, Visual CUT would simply overwrite the pdf file with its new linearized version. For example:
   …**"PDF_Linearize:c:\tmp\Inv_{@N}.pdf>>>>Password"**
   Use that approach only if you have a way to recreate or restore the original file because the linearization process might fail. For example, while pdf exports from Visual CUT seem to go through the linearize process without a problem, I've seen some cases where pdf exports of the same report from Crystal Designer fail the linearize process.

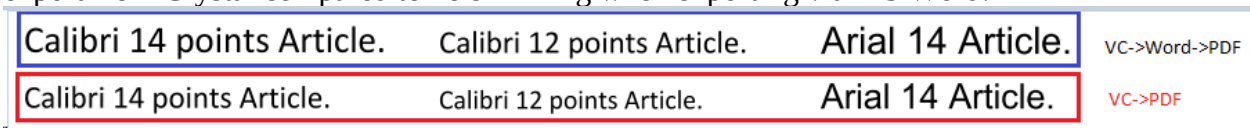
5. [Optional] **Password.** Leave blank if the file is not password protected. For example:
   …**"PDF_Linearize:c:\tmp\Inv_{@N}.pdf>>>>"**
   Or you may simplify this to:
   …**"PDF_Linearize:c:\tmp\Inv_{@N}.pdf"**

Notes:
As usual, any of these arguments can contain references to fields or formulas, and Visual CUT would dynamically replace the reference with the value in the report.

PDF_Linearize occurs after all other pdf operations such as **PDF_Merge** and **PDF_Protect**.

Export to PDF via MS WORD

When exporting to PDF, the Crystal runtime might **shrink** and **mishandle** certain fonts.
Exporting via Word fixes that issue and also adds options for generating **Tagged** pdf files and files that conform to the **PDF/A** standard.
Here is a comparison of two pdf exports, showing font shrinking problem (box in red) in PDF export from Crystal compared to no shrinking when exporting via MS Word:



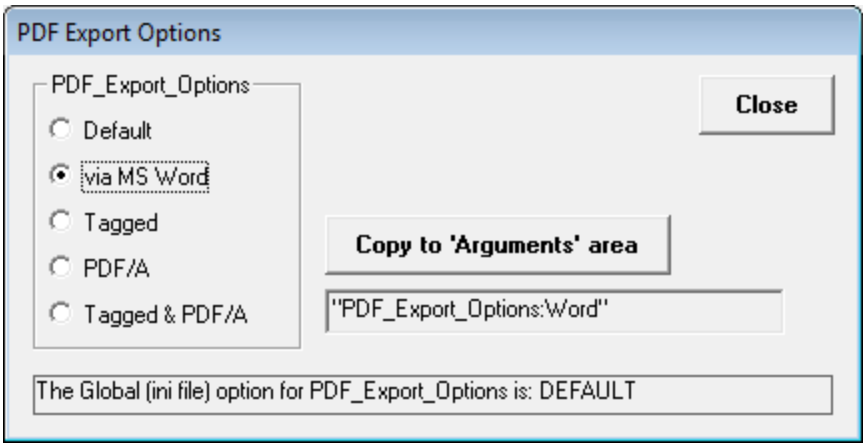
You may elect to default to PDF exporting via MS Word by using the Options dialog:

Alternatively, you can use a "PDF_Export_Options" command line argument to override the global (ini) setting. For example:
… **"PDF_Export_Options:Word"**

The options are:

1. "" or **Default** – Uses the Crystal Runtime to export
2. **Word** – Use MS Word to export to PDF (after internally exporting to MS Word)
3. **Tagged** – Same as 2, but export as a Tagged pdf file
4. **PDF/A** – Same as 2, but export as a PDF/A file
5. **Tagged and PDF/A** – Same as 2, but export as a Tagged PDF/A file

For pdf exporting format, the options button ... launches a **PDF Export Options** window showing the global (ini) setting, and providing choices for generating and saving the command line argument (if you need to override the global ini setting).



Notes:
If you specify Tagged or PDF/A options, the default behavior of setting PDF file properties and processing bookmark and link tags in the generated pdf file is skipped. This is because such post-processing would lose compliance with the PDF/A standard. So, unless you truly need a PDF/A file, you should use the lower-level options such as "Word" or the default Crystal Runtime pdf export.

The machine must have MS Word with ability to save to pdf.

Excel Functionality

Note: PC must have MS Excel installed.

Exporting to Excel 2007 (.xlsx) Files

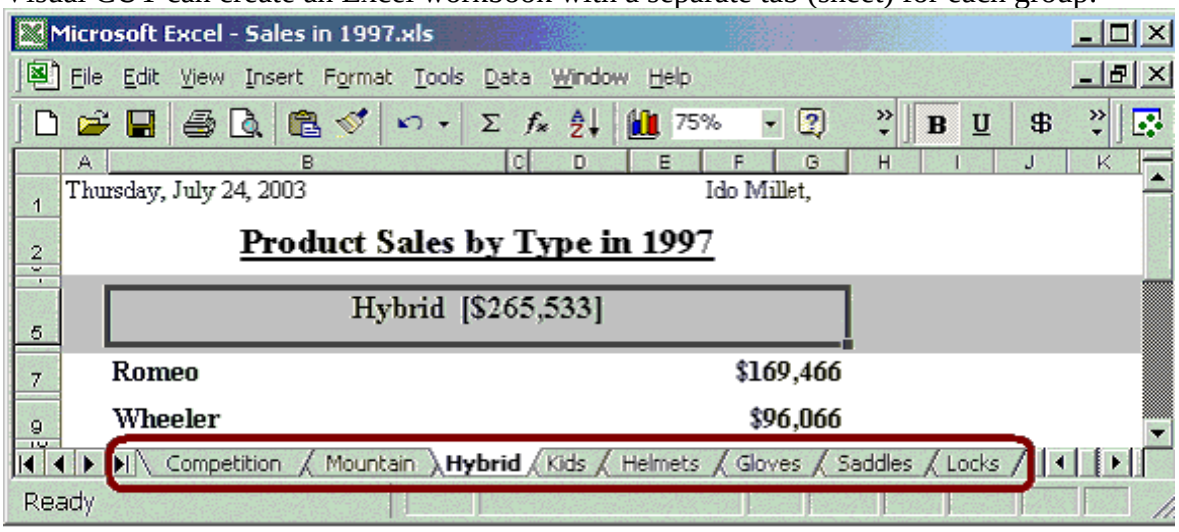
If you specify an export file name with a .xlsx file extension, even though the specified export format is Excel 97, Visual CUT takes care of producing an Excel 2007 .xlsx file.

The process actually starts by exporting to a temporary .xls file. If there are more than 65,536 rows in the export, the content will be split across multiple tabs inside the temporary .xls file. Visual CUT then converts the temporary .xls file to an Excel 2007 .xlsx file and merges all tabs into the first tab. The extra tabs are removed, and the temporary .xls file is deleted.

The overall effect is that when you specify a .xlsx file extension, you get an export to Excel 2007 format, supporting more than a million rows in a single tab and consuming much less disk space.

Combining Excel Bursting Exports into a Single Multi-Tab Spreadsheet

Visual CUT can create an Excel workbook with a separate tab (sheet) for each group:


Microsoft Excel - Sales in 1997.xls
File Edit View Insert Format Tools Data Window Help
Thursday, July 24, 2003
Ido Millet,
Product Sales by Type in 1997
Hybrid [$265,533]
Romeo $169,466
Wheeler $96,066
Competition Mountain Hybrid Kids Helmets Gloves Saddles Locks
Ready

Tip: right-clicking the VCR buttons to the left of the tab names launches a **tab navigation list**. This helps when the number or size of tab labels forces a scroll in order to see all of them.

The key to using this optional functionality is the naming of the export files:

- The name of the tab used for the individual sheets exported from each group is specified after the keyword '**Tab!**'
  (within the export file name and before the ".xls" file extension).
  Note: Visual CUT ensures tab names are acceptable to Excel by truncating long (>31) names and converting `:\/?*[]` characters to "legal" alternatives.

- The name of the spreadsheet file holding these individual sheets is determined by what's left when removing the 'Tab!sheet name' from the export file name.

For example, here is the export file name option used to generate the Excel file above:

c:\temp\Sales in {@Year} Tab!{Product_Type.Product Type Name}.xls

In this case, the highlighted area determines the sheet name inside the "master" spreadsheet and the rest of the text determines the "master" spreadsheet file name.
As usual, Visual CUT allows you to drag and drop any available fields or formulas to act as dynamic components in creating the file and sheet names.

If you want Visual CUT to e-mail the resulting file, then select the "Single Email" rather than the "Email for Each Group Level 1" option and specify the resulting file name in the Attach option.
For the case above, the resulting file name is: **c:\temp\Sales in 1997.xls**

Adding Excel Exports as Tabs in Existing/New Spreadsheets (Briefing Books)

Besides bursting a single report into multiple tabs in a single excel file, Visual CUT can also export a whole report or burst a report and add the resulting excel sheet(s) to existing excel files. This allows you to schedule several reports (typically using a batch file) and generate "briefing books" of several reports in multiple tabs within a single excel file. You would typically email the resulting excel file (at the end of the last scheduled step) to the recipients.
Note: you can use a Macro Enabled workbook (**.xlsm**) to host the tab, for further automation.

Let's assume you need to burst 10 reports to 100 agents. You would start the first report (about Commissions) with a burst into an export file name option such as:
c:\temp\Monthly Reports for {Agent_Name} in {@Month_Year} **TabInNewFile!**Commission.xlsx

This would result in 100 excel files named like:
Monthly Reports for **Richard Roper** in **April 2004**.xlsx
Monthly Reports for **Ido Millet** in **April 2004**.xlsx
etc.
Each of these workbooks will have a single tab (**Commission**) showing the Commission information for that agent. Note: as always, the tab name can also be dynamic by dragging & dropping fields/formulas to the area directly after **TabInNewFile**! key word (case sensitive).

You would then schedule the next report (about Policy Cancellations) to burst using an export file name option such as:
c:\temp\Monthly Reports for {Agent_Name} in {@Month_Year} **TabInOldFile!**Cancellations.xlsx

The **TabInOldFile!** ensures the **Cancellations** tab with the report information would be added to the excel workbook created in the previous step.

Note: When using **TabInOldFile!**, if the workbook file doesn't exist, it would be created.

You would continue in the same way with the other 8 reports (scheduled in that order in a single batch file. In the first 9 reports you will have no emailing but in the last one you will enable burst emailing so each agent gets their own "briefing book" as a multi-tab excel workbook instead of 10 separate excel files.

The scenario above assumes a bursting operation, but you can use the same exact approach in exporting whole reports to generate "briefing books".

Appending or Replacing Data for Existing Tabs

If **TabInOldFile!** encounters an existing tab, the export gets **appended** to the end of content in that tab. If you use **TabInOldFile_Replace!** Visual CUT inserts the tab if it doesn't already exist, but **replaces** the content of the tab if it already exists. This allows other sheets to refer to the content (periodically refreshed by Visual CUT) of that tab.

Note: **the file & tab naming conventions** that apply to this functionality are the same as described in the previous section.

Controlling Excel Tab Colors

When you specify tab names in excel exports via any of the methods described above, you can control the color of the excel tab by preceding the tab name with an RGB directive like this:

RGB225000000

The 3 sets of 3 digits following the RGB text specify the Red, Green, and Blue values (ranging from 0 to 255). See RGB color values at sites such as: http://web.njit.edu/~kevin/rgb.txt.html

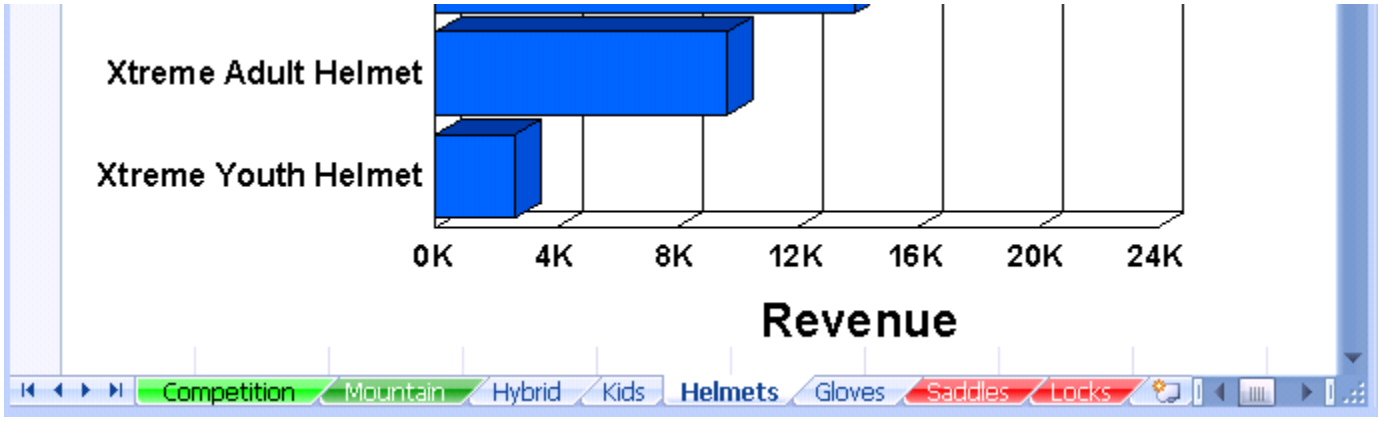
Visual CUT then takes care of coloring the excel tab according to the directive (and removing the directive from the tab name).

Here is an example of an export file name:
c:\temp\Sales Tab!**RGB225000000**{Product_Type.Product Type Name}.xls

The real power behind this feature is that the color directive can be a formula that dynamically sets the tab color based on performance as reflected in the report data. For example:
c:\temp\Sales Tab!**{@RGB_Tab_Color}**{Product_Type.Product Type Name}.xls

For example:

```
Select Sum ({@value}, {Product_Type.Product Type Name})
Case is > 1000000: "RGB000255000"
Case is > 300000:  "RGB000150000"
Case is > 11000:   ""
Case is <= 11000:  "RGB225000000"
default:       ""
```



Setting Up Print Properties for Excel Workbooks

You can instruct Visual CUT to change the print setup properties of an Excel workbook or a specified tab (sheet) inside that workbook. The command line argument structure is as follows:

… **"XLS_Print_Setup:Excel_File>Tab>…"**

The parameters (after the ":") are separated by a ">" and are as follows:

1. **Excel_File:** the full path and name of the excel file.
2. **Tab:** the tab (sheet) name. Use **1** to target the first tab. Leave **blank to target all tabs**.
3. **Orientation: Portrait**, **Landscape**, or leave **blank for no change**.
4. **N Pages Wide**: for example, use **1** to fit into 1 page wide. Leave blank for no effect.
5. **N Pages Tall**: for example, use **1** to fit into 1 page tall. Leave blank for no effect.
6. **Left Margin**: in inches. For example, **0.25**. Leave blank for no effect.
7. **Right Margin**: in inches. For example, **0.25**. Leave blank for no effect.
8. **Top Margin**: in inches. For example, **0.25**. Leave blank for no effect.
9. **Bottom Margin**: in inches. For example, **0.25**. Leave blank for no effect.
10. **Header Margin**: in inches. For example, **0.25**. Leave blank for no effect.
11. **Footer Margin**: in inches. For example, **0.25**. Leave blank for no effect.
12. **Center Horizontally**: **Y** for Yes. Any other value, including blank, for no effect.
13. **Center Vertically**: **Y** for Yes. Any other value, including blank, for no effect.
14. **Print Grid Lines**: **Y** for Yes. Any other value, including blank, for no effect.

Note: you can use field or formula names within the command line arguments.
The dynamic content of these fields/formulas would be substituted into the command line.

For example (all in one line):
"XLS_Print_Setup:c:\temp\{Customer.Country}.xls>>>1>1>>>>>>>>>"

The command line above would be typical of cases where you burst a report to multiple excel files, and you wish to fit the printout (when a user opens the file in Excel) into a single page.

Note: since there are 14 elements, **there should always be 13 instances of '>' separators**.

Auto Filter & Freeze Panes in Excel Exports

You can instruct Visual CUT to turn on the Auto Filter behavior within exported excel files.
The command line argument structure is as follows:
… **"XLS_AutoFilter:True"**
For more advanced options, use the following command line argument structure:
… **"XLS_AutoFilter:A1>>A2"**

The arguments are separated by >> and are as follows:
Spreadsheet Cell for Applying **Auto Filter** (any cell within a tabular range is a good target)
Spreadsheet Cell for Applying **Freeze Panes** -- Rows above and Columns to the left get frozen.
A2 freezes just the 1st row)

Note: leaving one of the two elements blank would result in processing only the non-blank argument

To process a file that is not the one being exported, specify the target file as the first argument:
… **"XLS_AutoFilter:c:\temp\test.xls>>True"**
or
… **"XLS_AutoFilter:c:\temp\test.xls>> A1>>A2"**

Column Auto Fit in Excel Exports

You can instruct Visual CUT to automatically fit the column widths in exported excel files. This is useful in "Data Only" exports. The command line argument structure is as follows:

… **"XLS_AutoFit:True"**

For more advanced options, use the following command line argument structure:

… **"XLS_AutoFit:MaxColumnWidth>>31>>False"**

The 3 elements after the ":" are separated by >> and are as follows:
MaxColumnWidth – fixed text
Maximum column width – a number between **1** and **255**
Wrap Text – True or False. Control whether columns adjusted down to the specified maximum width should wrap their text.

Password Protecting Excel Workbooks

You can instruct Visual CUT to protect any number of Excel files. The command line argument structure is as follows:

… **"XLS_PROTECT:Excel_File_List>Password"**

The parameters (after the ":") are separated by a ">" and are as follows:

1. **Excel_File_List:** comma separated list of the Excel files you wish to protect.
   If all files share the same folder, **you can specify the full path just for the first file**.
   If a file is not found, a warning is written to Failure.log and that file is skipped.
   Note: you can specify file names using **wild cards**.

2. **Password:** up to 15 characters in length.

Important Note: you can use field or formula names within the command line arguments. The dynamic content of these fields/formulas would be substituted into the command line. Among other things, this allows you to **easily protect individual Excel exports with different passwords for each group you are bursting.** For example (all in one line):

```
"XLS_PROTECT:c:\temp\{@Product}.xls>{@Product_Password}"
```

Protecting Excel Worksheets against User Viewing/Editing

As described at: http://spreadsheets.about.com/od/excel2007tips/qt/080428_lockcell.htm
Protecting data from change in Excel is a two step process.

1. locking/unlocking specific cells in your spreadsheet.
2. applying the *Protect Sheet* option. Until step 2 is completed, all data is vulnerable to change.

Visual CUT allows you to automate step 2 using the **XLS_Protect_Worksheets** command line argument.

Note: while **XLS_Protect** provides workbook access protection, **XLS_Protect_Worksheets** hides and protects cells from user editing.

You will typically use **XLS_Protect_Worksheets** after Visual CUT has populated an excel template worksheet with Crystal formula data (using **XLS_Range_Insert**).

The command line argument structure for **XLS_Protect_Worksheets** is as follows:

… **"XLS_Protect_Worksheets:Source_File>>Target_File>>Password"**

The parameters (after the ":") are separated by a ">>" and are as follows:

1. **Excel Source File** (path & name).

2. **[Optional] Excel Target File** (path & name). If none specified, source file is updated.

3. **Password:** up to 15 characters in length. If left blank, the user doesn't need a password to unprotect the worksheets.

You can use dynamic references to field or formula names within the command line argument. For example (all in one line):

… **"XLS_Protect_Worksheets:**`c:\temp\{@Branch}.xls>>{@Password}`**"**

Inserting File Exports into Excel Templates

Using **XLS_Range_Insert_File** command line argument, you can instruct Visual CUT to insert Excel (Data Only) exports into pre-formatted excel templates. This approach has several advantages compared to **XLS_Range_Insert** (see next section):

1. **XLS_Range_Insert** inserts data from formula values (limited to 65,534 characters). **XLS_Range_Insert_File** inserts data from files that can contain much more data
2. If the target cell is not empty, the content is appended to the first empty area below.
3. The method automatically detects the ***template row*** based on the target cell and the number of columns in the exported data. The format of all columns in the ***template row*** as well as the row height are applied to all inserted data.
4. You can elect to also clone content from all other columns in the template row. This allows you to **clone formulas that can refer to content in the inserted rows!**

Here is an example of how the command line argument is structured:
"**XLS_Range_Insert_File:**C:\TEMP\Template.xlsx>>C:\TEMP\Template_Filled.xlsx>>
c:\temp\export1.xlsx||Sheet1||B5||[Clone_Template_Row][Remove_Content_Top_Row]"

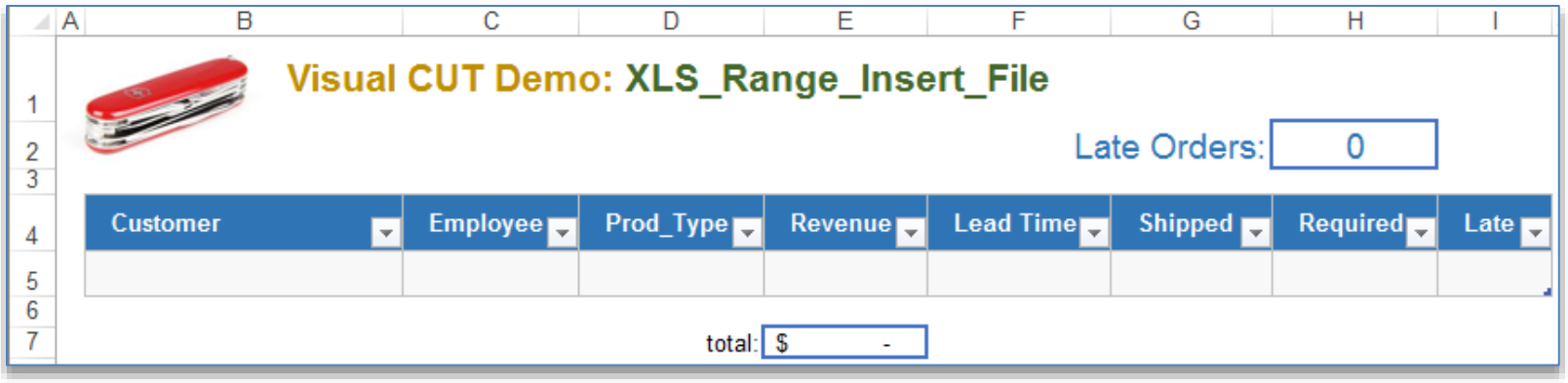
The elements (after the "**XLS_Range_Insert_File:**") are separated by a ">>" and are as follows:
1. Template excel file.
2. Resulting excel file. In append scenarios, can be same as template file
3. Data source & insert instructions consisting of 4 "||" delimited elements:
   a) Excel (data Only) export file to be inserted
   b) Target sheet in template
   c) Target cell in template
   d) Options (may leave blank):
      [Clone_Template_Row] is useful for cloning formulas outside template columns.
      [Remove_Content_Top_Row] skips the top row of the inserted data.

Notes:

a) If the template range is a Table with sort and filter conditions, these choices are reapplied after the insert, so the populated table is sorted and filtered just as in the template. See sample image: http://screencast.com/t/nsQiSnfhXK
b) If your template Table contains formula columns, sparklines, or any other content you wish to clone rather than overwrite, include a corresponding blank column in the excel export. You can use a formula that returns "" in Crystal to create such a blank column.
c) You can specify multiple data source & insert instructions separated by a "^^" delimiter.
d) Content below target cell gets pushed down as data is inserted.
e) Row height of target cell is cloned to all inserted rows
f) As always, the files path & name can contain dynamic references. For example: "XLS_Range_Insert_File:c:\myTemplate.xlsx>>{@Result}>>{@DataFile}||Sales||E8||"
g) You can fill templates in hidden sheets
h) At the end of the process, Visual CUT restores the focus in the workbook to the original sheet and original selections in all sheets.
i) The process can automatically handle cases where a Table contains a Summary Row.

The following page shows images demonstrating this functionality.

Sample Input & Output



The template workbook above contains conditional formatting for Revenue (Color Bars) and Lead Time (color scale from red to green) and is filled with data from an Excel (Data Only) export (with 7 columns) that looks like this:

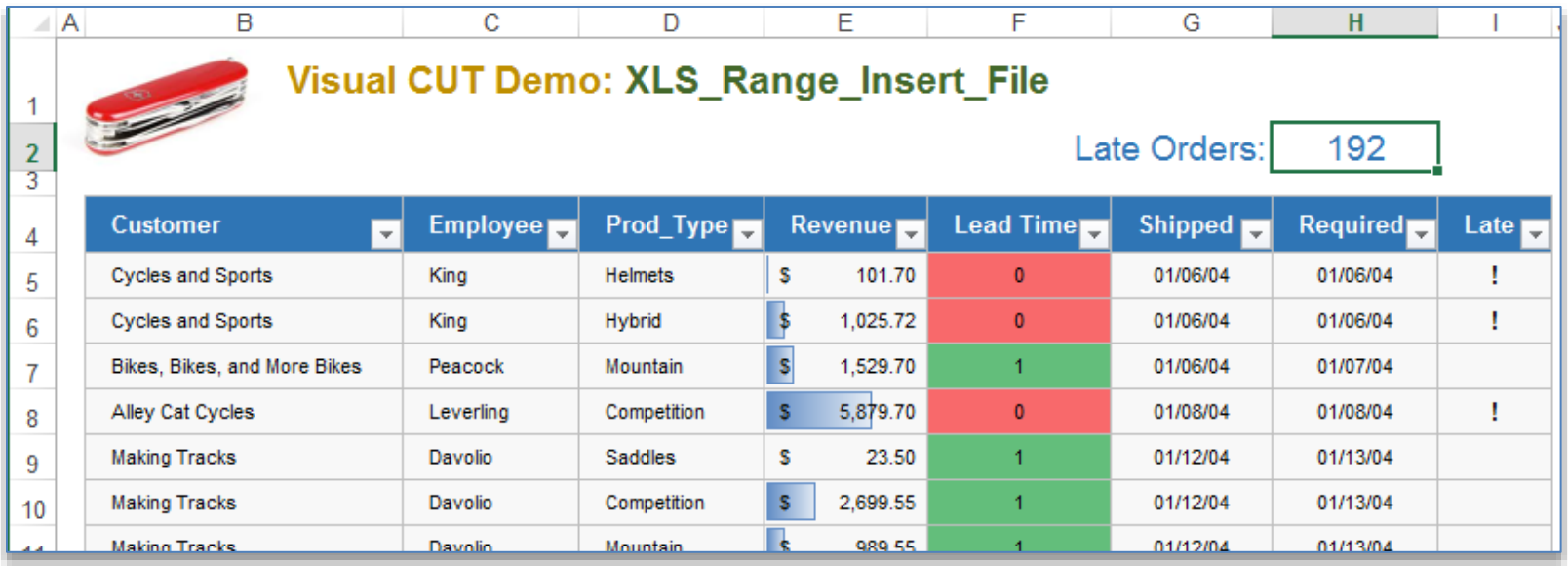
	A	B	C	D	E	F	G
1	Customer	Employee	Prod_Type	Revenue	LeadTin	Ship Date	Required Date
2	Cycles and Sports	King	Helmets	$102	0	01/06/2004	01/06/2004
3	Cycles and Sports	King	Hybrid	$1,026	0	01/06/2004	01/06/2004

While the Excel export contains 7 columns, the template workbook has an 8th column (column I) with a formula flagging late cases (Shipped date >= Required date).

Using the command line argument of:

```
"XLS_Range_Insert_File:C:\temp\Template.xlsx>>C:\Temp\Filled.xlsx>>
c:\temp\Data.xlsx||Sheet1||B5||[Remove_Content_Top_Row][Clone_Template_Row]"
```

the resulting workbook looks like this:



Inserting Crystal Values into Excel Templates

Using a command line argument, you can instruct Visual CUT to replace **named ranges** inside MS Excel spreadsheets with values of formulas with matching names. The file can then be saved to a new dynamically named file and emailed as part of a bursting Visual CUT process.

The advantage of this technique over directly exporting/bursting a Crystal report to excel spreadsheets is that you can **keep and control all elements in the template spreadsheet. For example, the template spreadsheet may have formulas, conditional formatting, hidden columns, and macros.**

Here's an example of the command line argument structure:
… **"XLS_Range_Insert:c:\template.xls>>c:\target.xls"**

The arguments (after the "**XLS_Range_Insert:**") are separated by a ">>" and are as follows:

1. The **path & name of the template XLS file** (containing named ranges).

2. [optional] The **path & name of the resulting XLS file**. If the command line specifies only a template file, that file will simply be updated.

As always, the files path & name can contain dynamic references. For example:
… "XLS_Range_Insert:C:\temp\Master.xls>>c:\temp**{Faculty.Faculty_Name}_{[yyyy]}**.xls"

Or, if the source file needs to be updated (without creating a new file:
… "XLS_Range_Insert:c:\temp**{Faculty.Faculty_Name}_{[yyyy]}**.xls"

Specifying Named Ranges in the Excel Template and Matching Formulas in Crystal

To create a named range in excel, simply select the cell(s) and type a name for the range in the cell address area above column A.

To identify a formula in a Crystal report that should provide the value(s) for a named range, the formula name must start with **VC_XLS_Range_Insert_** followed by the range name.
For example, the formula **VC_XLS_Range_Insert_Ref_J** would insert values into a named range called **Ref_J** .

Notes:
- Visual CUT can handle named ranges even if they reside inside hidden tabs.
- Crystal's formula string length is limited to 65,534 characters.
- In order for Visual CUT to recognize the value of the Crystal formula, it must be placed in:
  1. The Report Header or Footer or
  2. Group Header 1 or Group Footer 1 if the report is being burst by Visual CUT.

The named ranges must have a Workbook scope. If you create a named range and then make a copy of the sheet within the same workbook, you will need to use the Name Manager within Excel to rename and set the scope of the named range to 'Workbook'.

Crystal Formula Content for Multi-cell and multi-row Named Ranges

For a single cell named range, the Crystal formula can simply return the desired text.

For multiple cells in a single row named range, the formula value must be a strings delimited according to the following structure:
- Multiple cells should be separated with a "||" delimiter.
- Multiple rows should be separated with a "&&" delimiter

The named range doesn't need to have as many row cells as elements in the formula row. Visual CUT starts inserting cells in the left-most column of the named range and continues with as many cells there are in each formula row.

The named range the template Excel spreadsheet should have at least 2 rows if the formula will provide more than one row of data. Visual CUT will expand the named range to accommodate the number of rows.

Blank cell values should be provided as **[vc_null]**

Here is an example of this approach in the context of populating a named range (**Ref_J**) with faculty refereed journal publications. The final formula simply returns the value of a global string variable that has been accumulated via other formulas:

GH1 Formula to **reset** the global string variable:

```
WhilePrintingRecords;
Global stringvar Range_Values := "";
```

Detail Formula to **accumulate** the information into the Range_Values string variable:

```
WhilePrintingRecords;
Global stringvar Range_Values;
Local StringVar New_Range_Values;
Local stringvar ls_REF_J_Date;
IF IsNull({REF_J.Date}) Then
    ls_REF_J_Date := "[vc_null]"
else
    ls_REF_J_Date := {REF_J.Date};

New_Range_Values :=
{REF_J.Title} & "||" &
{REF_J.Journal Name} & "||" &
ls_REF_J_Date & "||" &
"[vc_null]" & "||" &
ls_REF_J_Points ;

IF Len(Range_Values) = 0 Then
Range_Values := New_Range_Values
Else
Range_Values := Range_Values & "&&" & New_Range_Values;
```

Group Footer Formula: {@VC_XLS_Range_Insert_Ref_J}
with a name matching the named range in the spreadsheet:

```
WhilePrintingRecords;
stringvar Range_Values;
```

Populating Multiple Named Ranges

To populate multiple named ranges, you simply need to have multiple formulas in the Crystal report. Each formula must be named (as explained above) to match the desired named range in the excel spreadsheet.

Visual CUT loops through all named ranges in the excel workbook and, if it can find a matching Crystal report formula, it will undertake populating the range with the formula value(s).

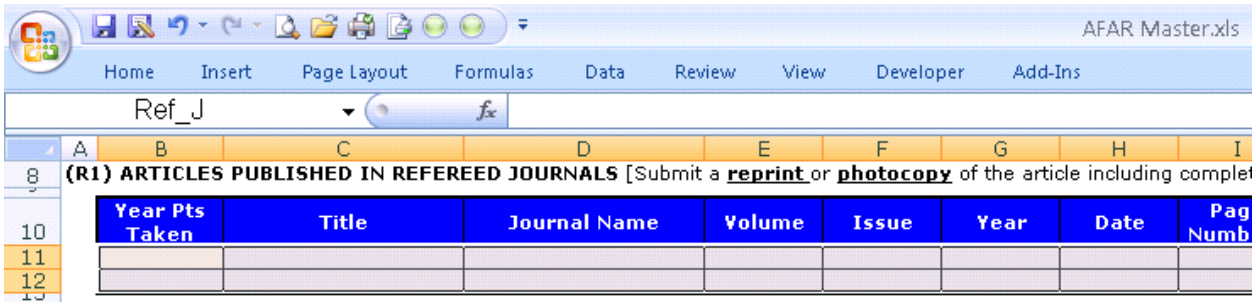
In some cases, the data for different named ranges may require formulas in different Crystal reports. You can simply create a multi-line batch file that calls one report and populates some named ranges, and then calls another report to populate other named ranges.

Notes:

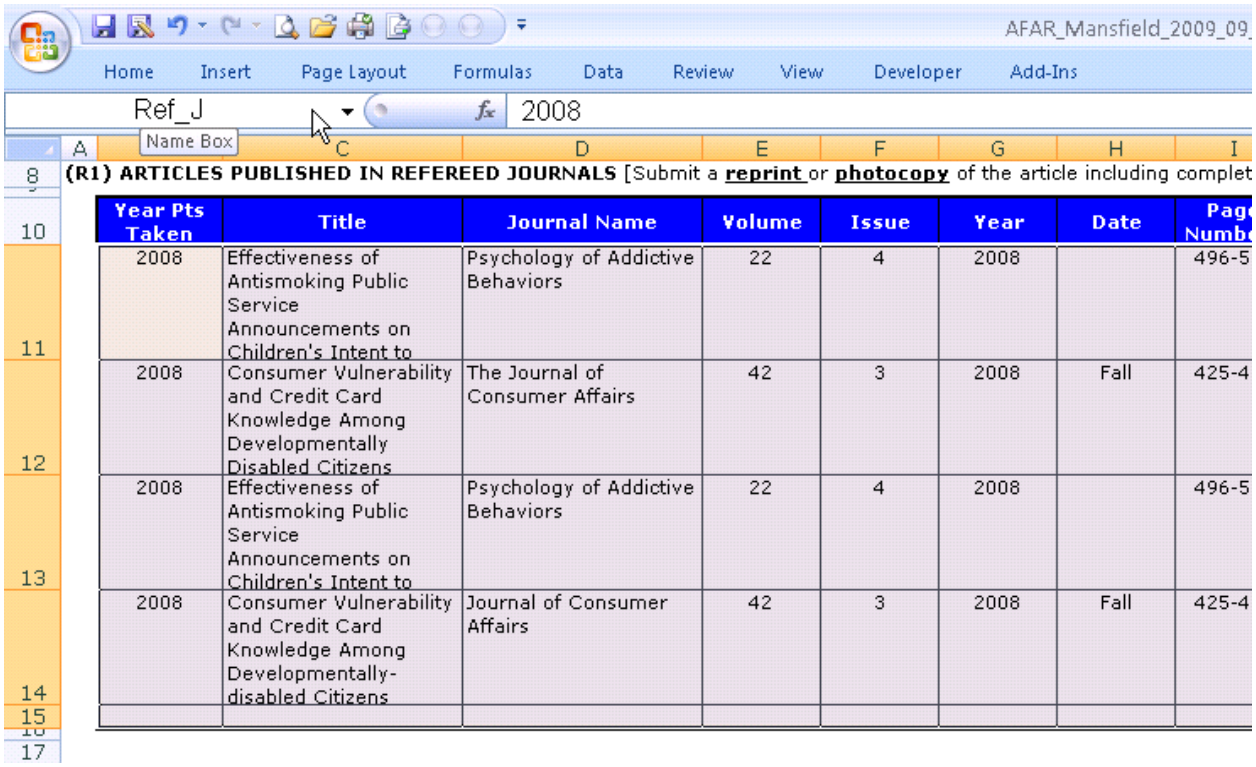
- You can specify the final target file as the email attachment since the **XLS_Range_Insert** processing occurs before emailing.
- If you wish to protect the resulting file from user editing, you may also use **XLS_Protect_Worksheets** described in a previous section.
- After the process of populating the ranges, Visual CUT sets the resulting workbook to open to the original sheet and selection that were last set in the source workbook.

Before & After Images

Here is what a portion of a template spreadsheet may look like with a named range (Ref_J) selected:



And here is that same range in the resulting excel file after Visual CUT automatically populated and expanded it with data from the **{@VC_XLS_Range_Insert_Ref_J}** Crystal formula:



Transferring Tabs To Another Workbook

Using a command line argument, you can instruct Visual CUT to transfer tabs from one workbook to another and even rename the transferred tabs based on Crystal fields/formulas. This is particularly useful when bursting report information into named ranges in a template workbook, and then gathering the resulting tabs into a single workbook with multiple tabs.

Here's an example of the command line argument structure:
… **"XLS_Transfer_Tabs:[c:\my.xls||Tab1::Tab2]>>[c:\{@WB}.xls||{@Tab1}::{@Tab2}]"**

The arguments (after the **XLS_Transfer_Tabs:**) are grouped into two main parts, each surrounded by square brackets and separated by a ">>":

1. The first part specifies the source excel workbook and, after a || delimiter, the list of tab names (separated by :: from each other) that should be transferred to the target workbook.

2. The second part specifies the same information, but for the target workbook.

Notes:

a) The target tabs can specify different names than the source tabs. Renaming occurs based on the position within the list of tabs.
b) If the source workbook doesn't exist, it gets created.
c) If a target tab already exists in the target workbook, it gets replaced.
d) Since this argument executes after XLS_Range_Insert, the two processes can be chained into one command line.
e) As always, the files path & name can contain dynamic references.

Splitting Excel Workbooks

Using a command line argument, you can instruct Visual CUT to split an Excel workbook so each tab becomes a separate PDF or Excel file.

Here's an example of the command line argument structure:
… **"XLS_Split_Tabs:c:\my.xlsx||PDF||c:\new_[Tab_Name].pdf||Landscape||True"**

The parameters (after the "**:**") are separated by a "**||**" and are as follows:

1. **Source_WorkBook:** path & name of the source excel file (e.g. c:\temp\Sales.xls)
2. **Target File Format:** PDF or XLSX
3. **Target File:** the path and name of the target file for each tab. The **[Tab_Name]** token gets replaced with the name of the tab.
4. **PDF Page Orientation: Landscape** or **Portrait**
5. Ignore Print Areas: **True** (include all content) or **False** (include Print Areas Content)

Notes:

a) Blank/Hidden tabs and skipped
b) If a target file for a given tab already exists, it gets replaced
c) Missing folders for target files are created automatically
d) For PDF format, the content of each tab gets fitted into a single PDF page
e) As always, the argument may contain dynamic references to fields/formulas.
for example:
…"XLS_Split_Tabs:**c:\temp\Sales_in_{@Year_Parameter}.xlsx**||PDF||
c:\temp\Sales_in_{@Year_Parameter}_for_**[Tab_Name]**.pdf||Landscape||True"

Generating Excel Pivot Tables

Visual CUT can automatically generate an Excel Pivot Table based on an existing or newly exported (Data Only) excel worksheet. This can be very valuable in cases where you want to give users an easy and familiar way to Slice & Dice the data.
The command line argument structure is as follows:
… **"XLS_Pivot_Table:Source_WorkBook>New_WorkBook>Tab>…"**

The parameters (after the ":") are separated by a ">" and are as follows:

1. **Source_WorkBook:** path & name of the source excel file (e.g. c:\temp\Sales.xls)
2. **New_WorkBook:** path & name of file to create (e.g. c:\temp\Sales_and_Pivot.xls**x**)
   - if left blank, the Pivot table gets added as 1st tab in the existing WorkBook
   - if the style argument is not blank, the destination file should have .**xlsx** extension and the machine should have **Excel 2007 or later**.
3. **Tab:** the sheet name in the source workbook where the data block is. Typically: **Sheet1**
4. **Range Name:** [Optional]. If blank, data with column headers is assumed to start at A1.
5. **Pivot/Tab Name:** Pivot Table and New Worksheet Tab. To Target an existing Tab, specify: PivotName||TabName||Cell_Address (for example, Pivot1||Sales||D20)
6. **Row Elements**: field names separated by "||". For example, "Country||City" would use Country as level 1 row group and City as level 2 row group
7. **Row Element Sort**: a sort code for each row element specified above, separate by "||"
   - **A** for **A**scending, **D for D**escending, **N** for **N**o Sort
   - **T#** for **T**op**N** (for example **T7** for Top 7), **B#** for **B**otton**N** (**B12** for Bottom 12)

   Last element (after "||") must specify the CAPTION name of the Column Controlling the Sort (e.g., "Revenue"). For example: "**T5||D||Revenue**" would show Top 5 countries and within that cities in Descending order based on "Revenue"
8. **Column Elements**: field names separated by "||". ("Employee||Product Name")
9. **Column Element Sort**: same logic as for Row Element Sort
10. **Page Filter Elements:** field names separated by "||". ("Product Class||Product Type")
11. **Data Elements**: separated by "||" and providing 4 sub-elements separated by "::"
    1. **Field name** – the column header from the raw data block (e.g. "Value")
    2. **Caption** - a user-friendly name. For example, "Revenue"
    3. **Summary Type** – current options include: Sum, Count, Average, Max, Min, or Var
    4. **Format String** – for example: $#,##0,K or **0.0%**

    For example: "**Value::Revenue::Sum::$#,##0,K||Late::% Late::Average::0.0%**" specifies 2 pivot table metrics. The 'Revenue' metric is a sum of the ***Value*** field, and is formatted as thousands of dollars. The '% Late' metric is an average of the ***Late*** field and is formatted as percent.
12. **Show Grand Totals**: "**Both**", "**None**", "**Rows**", or "**Columns**"
13. **Show Subtotals**: "**None**", "**Bottom**", or "**Top**"
14. **Style**: same style names as in Excel, but without spaces (e.g. "**PivotStyleMedium23**"
15. **Show Bands**: "**Both**", "**None**", "**Rows**", or "**Columns**"
16. **Show Blank Rows**: "Yes" to insert a blank line after each group level 1 row.
17. **Hide Data Tab**: "**No**", "**Yes**" (allow unhide), "**YES**" (no GUI option to unhide).
18. **Hide Field List**: Yes hides the field list on the right side of the screen.
19. **Data Orientation**: (optional) "**Rows**" **(Default)** or "**Columns**" (to show metrics side-by-side)
20. **Options** (optional): various options separated by '||' (e.g. Report_Layout=Tabular)

Let's take as an example the following command line argument (all in 1 line):

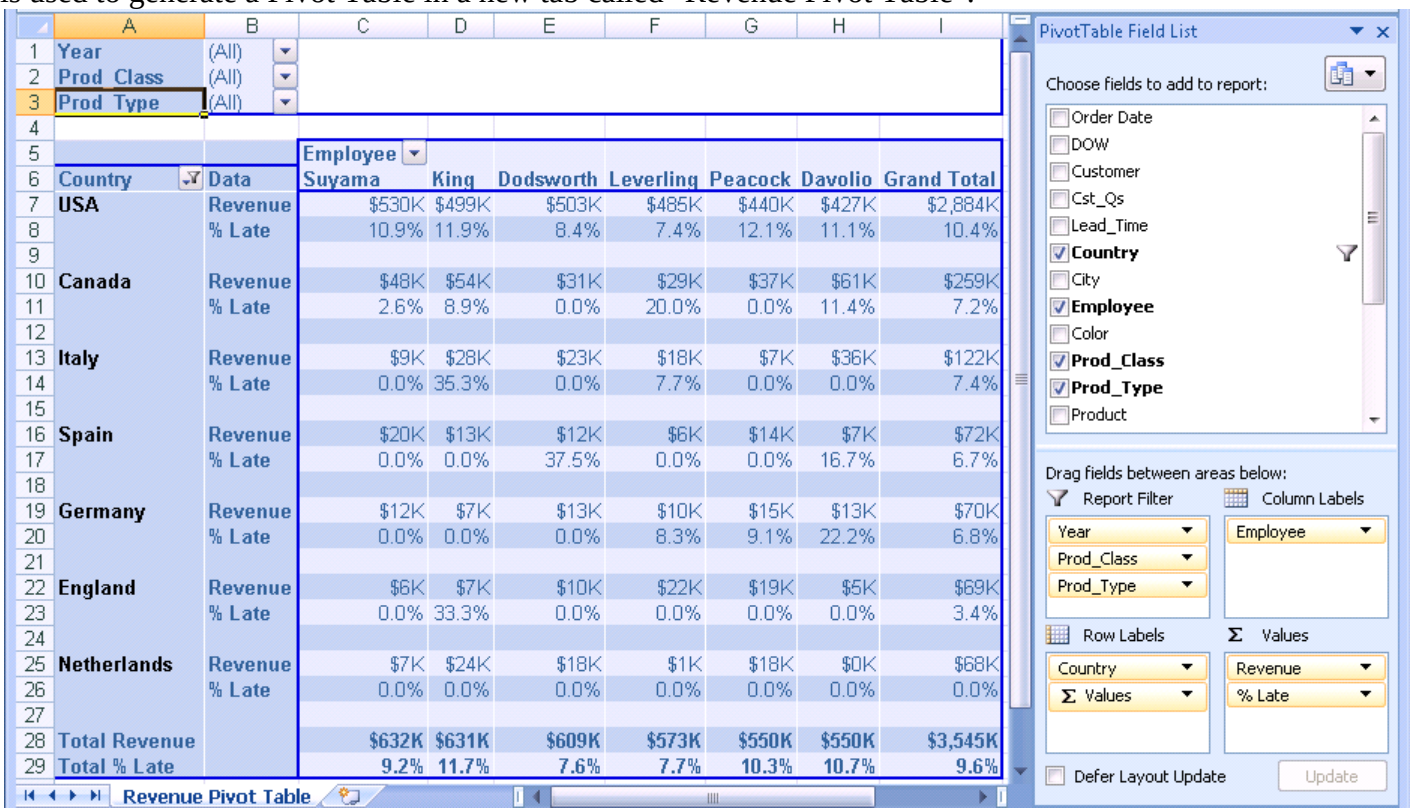
```
"XLS_Pivot_Table:c:\temp\Before_Pivot.xls
>c:\temp\After_Pivot.xlsx>Sheet1
>>Revenue Pivot Table
>Country>T7||Revenue>Employee>D||Revenue>Year||Prod_Class||Prod_Type
>Value::Revenue::Sum::$#,##0,K||Late::% Late::Average::0.0%
>Both>Bottom>PivotStyleMedium23>Rows>Yes>YES>False"
```

That command line argument tells Visual CUT to take the "Before_Pivot.xls" file (which was generated by Visual CUT as an Excel Data Only export) and convert it to a new "After_Pivot.xlsx" excel 2007 file. The data found in the Sheet1 tab

	Order Date	DOW	Customer	Cst_Qs	Lead_Time	Country	City	Employ	Color	Prod_Class	Prod_Type	Product	Quantity	Value	Supplier	Late	Year	Quarter	Month
2	09/30/2004	Thu	City Cyclist	6	11	USA	Sterling	Leverling		Accessory	Locks	Xtreme T	2	76	Craze	0	2004	3	9
3	01/26/2005	Wed	City Cyclist	8	1	USA	Sterling	Davolio	steel sa	Bicycle	Competitior	Descent	1	2,940	Craze	0	2005	1	1

Sheet1

is used to generate a Pivot Table in a new tab called "Revenue Pivot Table":



Year	(All)							
Prod_Class	(All)							
Prod_Type	(All)							
		Employee						
Country	Data	Suyama	King	Dodsworth	Leverling	Peacock	Davolio	Grand Total
USA	Revenue	$530K	$499K	$503K	$485K	$440K	$427K	$2,884K
	% Late	10.9%	11.9%	8.4%	7.4%	12.1%	11.1%	10.4%
Canada	Revenue	$48K	$54K	$31K	$29K	$37K	$61K	$259K
	% Late	2.6%	8.9%	0.0%	20.0%	0.0%	11.4%	7.2%
Italy	Revenue	$9K	$28K	$23K	$18K	$7K	$36K	$122K
	% Late	0.0%	35.3%	0.0%	7.7%	0.0%	0.0%	7.4%
Spain	Revenue	$20K	$13K	$12K	$6K	$14K	$7K	$72K
	% Late	0.0%	0.0%	37.5%	0.0%	0.0%	16.7%	6.7%
Germany	Revenue	$12K	$7K	$13K	$10K	$15K	$13K	$70K
	% Late	0.0%	0.0%	0.0%	8.3%	9.1%	22.2%	6.8%
England	Revenue	$6K	$7K	$10K	$22K	$19K	$5K	$69K
	% Late	0.0%	33.3%	0.0%	0.0%	0.0%	0.0%	3.4%
Netherlands	Revenue	$7K	$24K	$18K	$1K	$18K	$0K	$68K
	% Late	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Total Revenue		$632K	$631K	$609K	$573K	$550K	$550K	$3,545K
Total % Late		9.2%	11.7%	7.6%	7.7%	10.3%	10.7%	9.6%

Revenue Pivot Table

PivotTable Field List

Choose fields to add to report:

Order Date, DOW, Customer, Cst_Qs, Lead_Time, **Country**, City, **Employee**, Color, **Prod_Class**, **Prod_Type**, Product

Drag fields between areas below:

Report Filter: Year, Prod_Class, Prod_Type

Column Labels: Employee

Row Labels: Country, Σ Values

Values: Revenue, % Late

Defer Layout Update — Update

Country>T7||Revenue set the rows to Top 7 Countries by Revenue
Employee>D||Revenue set the columns to Employees sorted Descending by Revenue
Year||Prod_Class||Prod_Type set the Page filters (top-left drop-downs)
Value::Revenue::Sum::$#,##0,K||Late::% Late::Average::0.0% set Revenue & % Late as metrics
PivotStyleMedium23>Rows>Yes set the style, banded rows, and a space after each row group
The YES>False at the end hides the tab for the raw data and doesn't hide the field list.

Notes:

1. Visual CUT takes care of auto-fitting column widths for both the raw data block for the pivot table as well as for the pivot table itself.

2. If the information is sensitive, you can also password protect the workbook using a command line argument such as: "XLS_Protect:c:\temp\After_Pivot.xlsx>sesame"

3. **XLS_Pivot_Table** is always processed before **XLS_Protect** so you can generate the workbook with the pivot table, password-protect it, and email it using a single Visual CUT process.

4. To support Style & Banded effects, the Visual CUT machine must have Excel 2007 or later.

5. The last **Options** argument currently supports only 3 **Report_Layout** directives: Report_Layout=**Compact** / Report_Layout=**Outline** / Report_Layout=**Tabular** **Tabular** uses field names instead of generic 'Row Labels' and 'Column Labels.'

6. The "Top" value for the 'Show Subtotals' option applies only to cases where the Pivot Table has a single Data element. It also applies a compact Outline layout to the resulting Pivot Table.

7. Sample report for exporting (to Excel Data Only) as a basis for generating a pivot table is available for download at: http://www.milletsoftware.com/Download/Pivot_Data.rpt

8. To handle excel exports that require more than 65,536 rows of data, you can **simply specify a file extension of .xlsx** (instead of .xls). For other situations (where the excel file is not exported from Visual CUT), you may use the **XLS_to_XLSX** command line argument described in the next section.

9. You may specify multiple "XLS_Pivot_Table:…" arguments and Visual CUT will execute all of them in the order they are specified. If you wish to hide the data tab, set that option to Yes only on the last process that needs access to it.

Refreshing Data (Queries, Pivot Tables) from all Connections

Excel workbooks can use external data for Queries and Pivot Tables. You may need to automate the process of refreshing that data (causing all connections used by the workbook to retrieve the data from external sources such as databases or web services).
The command line argument structure is as follows:
…"XLS_Refresh:Source_WorkBook>>New_WorkBook>>Sleep_Seconds"
Or, if you don't wish to save the updated source file:
…"XLS_Refresh:Source_WorkBook>>>>Sleep_Seconds"
Or, if you if wish to save the updated source file and there's no need to wait for the data retrieval to complete (the query option of 'Enable Background Refresh' is disabled):
…"XLS_Refresh:Source_WorkBook"

The parameters (after the ":") are separated by a ">>" and are as follows:

1. **Source_WorkBook:** path & name of the source excel file (e.g. c:\temp\Sales_Before.xls)
2. **New_WorkBook:** path & name of file to create (e.g. c:\temp\Sales_After.xlsx)
3. **Sleep_Seconds:** number of seconds to wait after initiating the refresh (before saving)

Note: if New_Workbook is left blank, the process saves the source workbook after the refresh.

Running an Excel Macro

After exporting a report to Excel, you may wish to trigger a global excel macro (a macro that was created as logic that can apply to any open workbook) to process the generated data. Or perhaps you used ***TabInOldFile_Replace!*** to insert the excel export as a tab inside an existing workbook that has a Macro you wish to trigger.

The command line argument structure is as follows:
…
"XLS_Run_Macro:Source_WorkBook>>New_WorkBook>>Macro_WorkBook>>Macro"
Or, if you don't wish to save the updated source file:
"XLS_Run_Macro:Source_WorkBook>>>>Macro_WorkBook>>Macro"

If the Macro is available through an add-on or resides in the source workbook, leave the Macro_Workbook argument blank:
"XLS_Run_Macro:Source_WorkBook>>New_WorkBook>>>>Macro"
Or, if there is no saving to a target file):
"XLS_Run_Macro:Source_WorkBook>>>>>>Macro"

The parameters (after the ":") are separated by a ">>" and are as follows:

4. **Source_WorkBook:** path & name of the source excel file (e.g. c:\temp\Sales_Before.xls)
5. **New_WorkBook:** path & name of file to create (e.g. c:\temp\Sales_After.xlsx)
6. **Macro_WorkBook:** path & name of the workbook containing the Macro
7. **Macro:** the name of the Macro to run

Notes:

1. You may need to add the location of the source excel file to the list of trusted locations (using Excel's Trust Center).
2. If New_Workbook is left blank, the process does not save the source workbook after the process. So if you want to save the updated workbook to the same path and file name, you must specify it.
3. As usual, you can embed field/formula references in the command line argument. For example,
   "XLS_Run_Macro**:{@Exported}**>>>>C:\Excel\Global_Macros.xls>>MyMacro"

Replacing Content in Excel Files

Using a command line argument, you can instruct Visual CUT to replace any number of strings in an Excel file with specified substitutions. **The replacement logic can use dynamic values** from a Crystal field or formula. Also, by forcing a re-evaluation of the content in cells with replaced content, **exported content in a cell can be turned into a dynamic Excel formula**. The command line argument structure is as follows:

… **"XLS_Replace:InFile||OutFile||find1>>replace1::find2>>replace2||Options"**

The parameters (after the ":") are separated by a "||" and are as follows:

1. **InFile:** the file path & name for the source Excel file.

2. **OutFile:** the file path & name for the resulting Excel file.
   - If no OutFile is specified, the source file gets updated
   - if the target folder doesn't exist, Visual CUT creates it
   - if no path is specified (just name) the path of the source file is used

3. **Find & Replace Pairs:** unlimited number of find and replace elements.
Each "find" element is separated from its "replace" element by a '**>>**'.
Each pair is separated from the next pair by a '**::**'
   - To specify special string characters, such as Carriage Return or Line Feed, use Chr() expressions (e.g., **Chr(10)** or **Chr(13)**.
   - To remove strings, specify them in the 'find' and leave the 'replace' as blank

4. **Options:** Leave blank. May be used for future enhancements.

For example, to replace **^=** with **=** (useful for turning formula expressions into formulas, if in Crystal you export the expression starting with "**^=**", use this command line argument:

… **"XLS_Replace:c:\temp\Input.xlsx||||^=>>=||"**

Dynamic Field Names
You can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. The dynamic content of these fields/formulas would be substituted into the command line argument.

Exporting Formula Expressions to Excel, and Activating Them

Imagine you wish to export a Crystal report to Excel, and some of the cells in the export contain formula expressions like this: "**=ROUND(C3*(1+($D$2/100)),2)**"
The excel export would treat such cells as TEXT rather than as a formula that refers to other cells in the exported file. To activate the cell so it becomes an Excel formula, change the expression in Crystal so it starts with something unique. For example: "**^=ROUND(C3*(1+($D$2/100)),2)**"
Then, use **"XLS_Replace:c:\temp\MyFile.xlsx||||^=>>=||"** to activate the formula.

Convert XLS Files to XLSX (and merging sheets)

Using **XLS_to_XLSX** command line argument, you can ask Visual CUT to convert a file from .xls to .xlsx format. The key benefits are:

- .xlsx files are **not limited to 65,536 rows per sheet**. During exports to .xls, Crystal and Visual CUT create multiple tabs inside the workbook when there are more than 65,536 rows. **The conversion to .xlsx can merge these multiple tabs into one tab**.
- **.xlsx files are smaller**. For example, an export to .xls that created two tabs (one tab with 65,536 rows and one tab with 56,027 rows) resulted in a **11.3MB .xls** file. After converting and merging to a single-tab .xlsx file (with 121,563 rows), the file size dropped to just **2.7MB**.

The command line argument structure is as follows:

… **"XLS_to_XLSX:Input_xls_File>>Output_xlsx_File>>Merge_Tabs?"**

For example: "XLS_to_XLSX:c:\temp\out.xls>>c:\temp\out.xlsx>>True"

The parameters (after the ":") are separated by a ">>" and are as follows:

1. **Input_xls_File:** path & name of the source .xls file (e.g. c:\temp\Sales.xls)
2. **Output_xlsx_File:** path & name of .xlsx file to create (e.g. c:\temp\Sales.xls**x**)
3. **Merge_Tabs?:** use a value of ***True*** to request that all tabs be merged into the first tab (tabs 2 and later are then removed).

Notes:

1. **If you simply specify an export file name with an.xlsx extension, Visual CUT executes an implicit XLS_to_XLSX conversion** (it exports to a temporary .xls file, converts and merges sheets into the specified .xlsx file, and deletes the temporary .xls file. This means you do not need to explicitly specify an XLS_to_XLSX argument.
2. The Visual CUT machine must have Excel 2007 or later.
3. To ensure proper merging of multiple tabs, make sure the first column in the excel export is not blank.
4. **XLS_to_XLSX** is executed before **XLS_Pivot_Table**. This allows a single command line to trigger exporting of a large data set to a multi-tab .xls file, conversion to a single-tab .xlsx file, generation of a pivot table, and emailing of the resulting workbook.

Convert Excel Files to PDF

Using a command line argument, you can instruct Visual CUT to save an Excel file to a PDF File. Here's an example of the command line argument structure:

… **"XLS_Save_As:c:\temp\Invoice.xls>c:\temp\Invoice.pdf>PDF>0>1>Competition"**

The parameters (after the "**XLS_Save_As:**") are separated by a "**>**" and are as follows:

1. The **path & name of the Excel file (xlsx files are supported as well)**
2. The **path & name of the target PDF file**
3. **Save As Format (PDF)**
4. **Optimized For Screen (1/0)**
   a value of **0** would create a higher-quality (optimized for print) but larger pdf file.
5. **Ignore Print Areas: 1**-True, 0=False
6. **Save Scope: Tab Name, or 0**=Workbook, or Tab Number: **1**=1st worksheet, **2**=2nd, …
   note: currently, only a single value between 0 and 9 is acceptable,
   so **you can save either the whole workbook or one of the first 9 tabs**.

Notes:

If you wish to fit the tab content into a single pdf page, you may need set the Print properties for the tab to **'Fit Sheet on One Page'**.

As usual, any of these arguments can contain references to fields or formulas and Visual CUT would dynamically replace the reference with the value in the report. For example:
… **"XLS_Save_As:{@file_name}.xlsx>{@file_name}.pdf>PDF>0>0>1"**

The machine must have MS Excel 2007 or higher. If 2007, it must have the "2007 Microsoft Office Add-in: Microsoft Save as PDF or XPS AddOn" which allows Microsoft Excel to save documents as PDF. If you don't already have that option enabled in MS Excel, you can download it from:
http://www.microsoft.com/downloads/details.aspx?FamilyId=4D951911-3E7E-4AE6-B059-A2E79ED87041&displaylang=en

Convert Excel Files to HTML

Using a command line argument, you can instruct Visual CUT to save an Excel file to a PDF File. Here are example of the command line argument structure:

to convert the Competition tab:
… **"XLS_Save_As:c:\temp\Invoice.xlsx>c:\temp\Invoice.htm>HTML>0>0>Competition"**
to convert a Pivot Table called Revenue, residing on a Revenue tab. No title:
… **"XLS_Save_As:c:\temp\After.xlsx >c:\temp\Revenue.htm>HTML>>Revenue>Revenue"**

"XLS_Save_As:c:\temp\After_Pivot.xlsx>:c:\temp\Revenue_PivotTable.htm>HTML>0>Revenue Pivot Table>Revenue Pivot Table"

The parameters (after the "**XLS_Save_As:**") are separated by a "**>**" and are as follows:

1. The **path & name of the Excel file (xlsx files are supported as well)**
2. The **path & name of the target HTML file**
3. **Save As Format (HTML)**
4. **Pivot Table Title:** Ignored if not targeting a Pivot Table. Set to blank to avoid a title.
5. **Pivot Table Name:** Leave as blank or 0 if not targeting a Pivot Table
6. **Save Scope: Tab Name, or 0**=Workbook, or Tab Number: **1**=1st worksheet, **2**=2nd, …
   note: currently, only a single value between 0 and 9 is acceptable,
   so **you can save either the whole workbook or one of the first 9 tabs**.

Notes:

The machine must have MS Excel installed.

A typical use scenario is to use Visual CUT to export & generate a Pivot Table, or fill a range or cell values in an excel template with formula values from a Crystal report. Then, embed the resulting HTML Save-As version of the populated excel file as an email message body using an [Insert_File:…]] directive.

As usual, any of these arguments can contain references to fields or formulas and Visual CUT would dynamically replace the reference with the value in the report. For example:
… **"XLS_Save_As:{@file_name}.xlsx>{@file_name}.htm>HTML>0>0>1"**
Or
… **"XLS_Save_As:c:\temp\After.xlsx >c:\temp\Revenue.htm>HTML>{@Pivot_Title}>Revenue>Revenue"**

Convert Excel Files to CSV

Using a command line argument, you can instruct Visual CUT to save an Excel file to a CSV File. Here are example of the command line argument structure:
… **"XLS_Save_As:c:\temp\Invoice.xlsx>c:\temp\Invoice.csv>CSV>>>1"**

The parameters (after the "**XLS_Save_As:**") are separated by a "**>**" and are as follows:

1. The **path & name of the Excel file (xlsx files are supported as well)**
2. The **path & name of the target CSV**
3. **Save As Format (CSV)**
4. **Leave blank**
5. **Leave Blank**
6. **Save Scope: Tab Name, or 0**=Workbook, or Tab Number: **1**=1st worksheet, **2**=2nd, …
   note: currently, only a single value between 0 and 9 is acceptable,
   so **you can save either the whole workbook or one of the first 9 tabs**.

Notes:
The machine must have MS Excel installed.

A typical use scenario is to use Visual CUT to **export a report that contains a subreport**. Such cases don't export well directly to CSV, but exporting first to Excel (Data Only) and then converting to CSV solves the problem.

As usual, any of these arguments can contain references to fields or formulas and Visual

MS Word Functionality

Note: PC must have MS Word installed.

Protecting MS WORD Files

Using a command line argument, you can instruct Visual CUT to:
a) allow viewing of a Word document only for user who knows the **Open Password**.
b) **restrict the type of editing** a user is allowed, unless the user knows the **Modify Password**.

A typical use scenario is when you have as a template document with form fields. Visual CUT takes care of populating content in this template document using **Word_Replace_Tags**. Before emailing the resulting file to a customer, you may need to restrict the customer to only form fields modifications.

Here's an example of the command line argument structure:

… **"WORD_Protect:c:\in.doc>>c:\out.docx>>OpenPass>>ModifyPass>>ModType"**

The parameters (after the "**WORD_Protect:**") are separated by a ">>" and are as follows:

1. The **path & name of the source file**
2. The **path & name of the Save-To file** (this can be same as the original file)
3. **Open Password required to view the file**
4. **Modify Password required to remove the modify restrictions**
5. **Modify Type** allowed for users without Modify Password. Valid options:
    a. **Allow_Only_Form_Fields** (user can only modify form fields
    b. **Allow_Only_Comments** (user can only add comments)
    c. **Allow_Only_Revisions** (user can only add revisions)

Notes:

as usual, any of these arguments can contain references to fields or formulas and Visual CUT would dynamically replace the reference with the value in the report. For example:
…**"WORD_Protect: {@in}>>{@out}>>Sesame>>mod123>>Allow_Only_Form_Fields"**

Inserting Crystal Values into MS Word Documents

Note: starting with Visual CUT 6.2002 (April 2010) this functionality is **also available from the Graphical User Interface** by simply selecting **Word_Replace_Tags** as the export format.

Using a command line argument you can instruct Visual CUT to replace references to Crystal field or formulas inside MS Word documents with values of these formulas. The file can then be saved to a new dynamically named file and emailed as part of a bursting Visual CUT process.

The advantage of this technique over directly exporting/bursting a Crystal report to a Word document is that Crystal exports to Word tend to have formatting and editing problems. By creating your own Word document to act as a template, you ensure precise formatting and problem-free editing.

Here's an example of the command line argument structure:

… **"WORD_Replace_Tags:c:\template.doc>c:\Contract_{Cust_Name}.doc>NoAppend"**

The parameters (after the "**WORD_Replace_Tags:**") are separated by a "**>**" and are as follows:

1. The **path & name of the original MS Word file** (containing references to Crystal formulas).

2. The **path & name of the target MS Word file** (after replacing references with values).

3. optional **Append** (or **NoAppend**) directive: if you specify "…>**Append**" at the end of the command line argument, the populated template document content will be appended to the end of the target file, if the target file already exists. If the target file doesn't already exist, the target file is created.

Specifying Field/Formula References (tags) in the Word document

If, for example, you wish to reference a Crystal formula called **{@Customer_Name}**, you should use the following text inside your MS Word document: **#{@Customer_Name}#** . In other words, simply enclose the formula name in # symbols.

Note that in order for Visual CUT to replace the reference with the value of the field/formula during processing, the field or formula must be placed in:
1. The Report Header or Footer if the report is processed Whole by Visual CUT, or
2. Group Header 1 or Group Footer 1 if the report is being burst by Visual CUT.

Notes:

- In bursting scenarios, you will want to target MS Word document path & name to be dynamic as in the command line argument example above (where the target file name contains {Cust_Name} as a dynamic portion).

- You can specify the target file as the email attachment since the processing of this command line argument occurs before emailing.

- Any dynamic field shown as available for drag & drop in Visual CUT can be referenced within the MS Word document. This includes the special date constants. Just remember to enclose the reference inside # signs. For example, #{[yyyy]}# would return the current year.

- There is no limit on the length of field/formula text that can be used to replace references. This means you can freely use data from memo or other large string fields.

Populating Word Tables with Crystal Formula Data

During **WORD_Replace_Tags** processing (see section above), you can instruct Visual CUT to also populate tables with data from Crystal formulas. To populate the 1st table you name the formula **Word_Replace_Table_1**. To populate the **N**th table in the document, name the formula **Word_Replace_Table_N**

Remember to place the formula in RH/RF section if not bursting or in GH1/GF1 section if the formula provides different data for each Group level 1 during bursting. As always, the formula or the section may be suppressed.

The formula should contain a strings delimited according to the following structure:
- Table Rows should be separated with a "][" delimiter.
- Table cells should be separated with a "||" delimiter

The table in the template Word document should already have at least as many rows as you expect to populate. Visual CUT will delete the extra rows. This way, any formatting you applied to the table would be maintained.

In most cases, the final formula would simply return the value of a global string variable that has been accumulated via other formulas. For example, if you are bursting the sample report Visual_Cut_11.rpt and wish to populate the 1st Table in a Word document with information about the Product Name (1st column) and Revenue (2nd Column), you may use the following 3 formulas:

GH1 Formula to **reset** the global string variable:

```
WhilePrintingRecords;
If NOT InRepeatedGroupHeader THEN Global StringVar Table_1 := "";
```

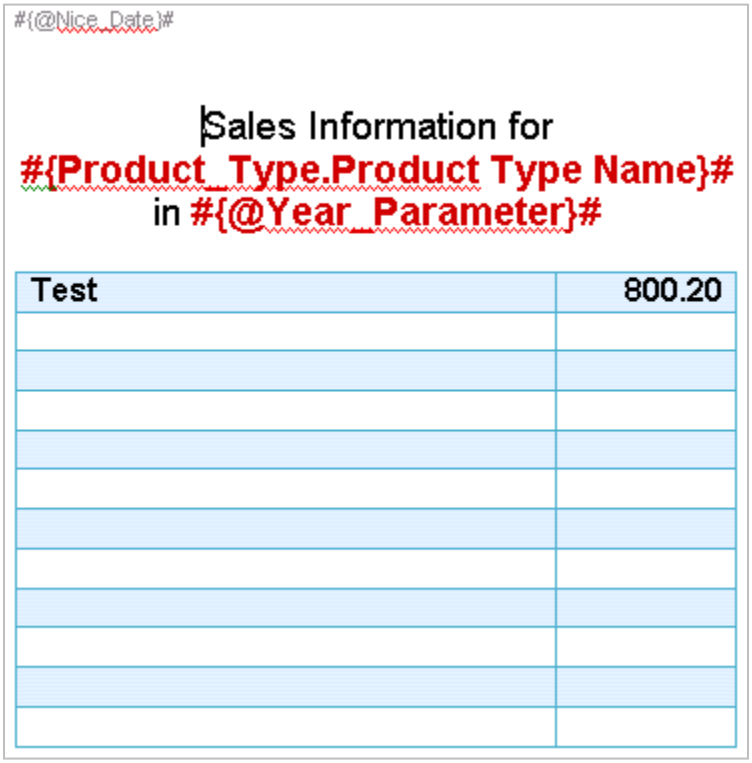
GH2 section Formula to **accumulate** the information into the Table_1 string variable:

```
WhilePrintingRecords;
Global stringvar Table_1;
IF Len(Table_1) = 0 Then
// First table row. Separate cell values with "||"
Table_1 := {Product.Product Name}+ "||" + "$" + ToText(Sum ({@value},
{Product.Product Name}),0)
ELSE
// Additional row, so add it to the accumulated string variable (Table_1)
// after a "][" to indicate a new row
Table_1 := Table_1 + "][" + {Product.Product Name}+ "||" + "$" + ToText(Sum
({@value}, {Product.Product Name}),0)
```

Group Footer Formula to **pass the string to Visual CUT**:

```
WhilePrintingRecords;
Global StringVar Table_1;
```

Using the above scenario, a template document (c:\temp\Template.doc) that looks like this:



Can be used via the following command line (all in 1 line):

```
"C:\Program Files\Visual CUT 11\Visual CUT.exe" -e "C:\Program Files\Visual
CUT 11\Visual_CUT_11.rpt"
"WORD_Replace_Tags:c:\temp\template.doc>c:\temp\{Product_Type.Product Type
Name}.doc>NoAppend"
```

In this case, you should set Visual CUT to burst this report. Any format would do since you would ignore the export and use the Word documents. Here's what Gloves.doc looks like:

Replace Formatting in MS WORD Files

Using a command line argument, you can instruct Visual CUT to find and replace a specified format throughout a Word document.

This functionality was developed to allow an export from Crystal to Word to result in Double Underline (typical for totals in accounting reports). Since Crystal doesn't have a double-underline, you can apply a Strikethrough in Crystal, and then convert it to Double Underline in the resulting MS Word export.

Here's an example of the command line argument structure:

… **"WORD_Replace_Format:c:\in.doc>c:\out.doc>Strikethrough>DoubleUnderline"**

The parameters (after the "**WORD_Replace_Format:**") are separated by a "**>**" and are as follows:

6. The **path & name of the source file**
   (typically, this would be the exported file but you can convert any WORD file).
7. The **path & name of the Save-To file** (this can't be same as original file)
8. **Format to search for** (currently, only ***Strikethrough*** is supported)
9. **Format to replace with** (currently, only ***DoubleUnderline*** is supported)

Notes:

as usual, any of these arguments can contain references to fields or formulas and Visual CUT would dynamically replace the reference with the value in the report. For example:
… **"WORD_Replace_Format:{@in}>{@out}>Strikethrough>DoubleUnderline"**

Printing MS WORD Files

Using a command line argument, you can instruct Visual CUT to print a specified MS Word file. Using field/formula references, you can dynamically set the file to be printed, the printer to be used, and the number of copies.

Here's an example of the command line argument structure:

… **"WORD_Print:{@Word_File}>>{@Printer_Name}>>{@Copies}"**
or
… **"WORD_Print:c:\temp\test.docx>>Default>>1"**

The parameters (after the "**WORD_Print:**") are separated by ">>" and are as follows:

6. The **path & name of the MS Word file**
7. The name of the destination **printer**, or **Default**
8. The number of copies

Notes:
as usual, any of these arguments can contain references to fields or formulas and Visual CUT would dynamically replace the reference with the value in the report.

Printing MS WORD Files with Watermarks

You can print exported Word files with dynamic watermarks for each printed copy (e.g. "Copy 1 of 4"). See sample image: http://screencast.com/t/oSaqXcqvidWF

If you also need to export to another file format, simply specify multiple export file names. For example, c:\temp\{@Invoice}.**pdf**;c:\temp\{@Invoice}.**doc**

The command line argument structure:
… **"WORD_Print_Watermark:{@Word_File}>>{@Printer_Name}>>{@Copies}>>**
WatermarkText>>Font>> Bold>>Italic>> FontSize>>Width>>Height>>
Rotation>>Transparency>>Options"

Example 1: (stretching to Watermark to length of 7.5 inches and height of 1.3 inches):
… **"WORD_Print_Watermark:c:\test.docx>>Default>>2>>Copy [[N]] of [[M]]>>**
Calibri>>False>>False>>12>>1.3>>7.5>>315>>0.5>>"

Example 2 (62 point font size without width/height stretching):
… **"WORD_Print_Watermark:c:\test.docx>>Default>>2>>Copy [[N]] of [[M]]>>**
Calibri>>False>>False>>62>> >> >>135>>0.5>>"

The parameters (after the "**WORD_Print_Watermark:**") are as follows:

1. The **path & name of the MS Word file**
2. The name of the destination **printer**, or **Default**
3. The number of copies
4. The Watermark text. Visual CUT replace [[N]] with the copy number and [[M]] with number of copies, so **Copy [[N]] of [[M]]** becomes Copy 1 of 4, Copy 2 of 4, etc.
5. **Font** (e.g. Calibri)
6. **Bold**: True or False
7. **Italic**: True or False)
8. **Font Size**: in points. Note: gets stretched if **Width** and/or **Height** are specified!
9. **Width**: in inches. Stretches text to desired width. Leave blank or space to ignore
10. **Height**: in inches. Stretches text to desired height. Leave blank or space to ignore
11. Rotation: Clockwise rotation in degrees. use 315 for diagonal, 0 for horizontal etc
12. Transparency: from 0 to 1. 0.5 = semi-transparency
13. Options: leave blank

Notes:
as usual, any of these arguments can contain references to fields or formulas and Visual CUT would dynamically replace the reference with the value in the report.

Convert WORD Files to PDF

Using a command line argument, you can instruct Visual CUT to save a WORD file to a PDF File. Here's an example of the command line argument structure:

… **"WORD_Save_As:c:\temp\Invoice.doc>c:\temp\Invoice.pdf>PDF>0>0>0>1>0"**

The parameters (after the "**WORD_Save_As:**") are separated by a "**>**" and are as follows:

14. The **path & name of the WORD file** (typically, this would be the exported file or the File you created using WORD_Replace_Tags, but you can convert any WORD file).
15. The **path & name of the target PDF file**
16. **Save As Format (PDF)**
17. **Optimized For Screen (1/0)**: a value of 0 would create a higher-quality (optimized for print) but larger pdf file.
18. **Create Bookmarks** (**0**=No, **1**=by Word Section Headings, **2**=by Word Bookmarks)
19. **Create Tagged (structured) PDF** (**1**=Yes, **0**=No)
20. **Substiture Bitmaps for Missing Fonts** (**1**=Yes, **0**=No)
21. **Use PDF/A Standard** (for archiving purposes) (**1**=Yes, **0**=No)

Notes:

as usual, any of these arguments can contain references to fields or formulas and Visual CUT would dynamically replace the reference with the value in the report. For example:
… **"WORD_Save_As:{@file_name}.doc>{@file_name}.pdf>PDF>0>0>0>1>0"**

The machine must have MS Word 2007 or higher. For Word 2007 you may need to download the "2007 Microsoft Office Add-in: Microsoft Save as PDF or XPS AddOn" from: http://www.microsoft.com/downloads/details.aspx?FamilyId=4D951911-3E7E-4AE6-B059-A2E79ED87041&displaylang=en

Text Functionality

Splitting Text Files by Embedded Tags

Using a command line argument, you can instruct Visual CUT to look for tags inside a text file and to split the file into multiple files. The process is faster than regular bursting and can be useful for generating many thousands of files.

Here's an example of the command line argument structure:

… **"TXT_Split_Tags:c:\temp\ToolTips.txt>>Replace"**

The parameters (after the "**TXT_Split_Tags:**") are separated by a ">>" and are as follows:

1. The **path & name of the text file you wish to split**
2. Desired action when split file already exists: **Replace** or **Append**

Within the text file, you should embed text tags that indicate the split file path & name that should apply for the following text. The structure and location of the tags should follow this example:

```
[[#Split_Tag::c:\temp\Some_File_Name.htm#]]<HTML>
…
…
</HTML>
[[#Split_Tag::c:\temp\Another_File_Name.htm#]]<HTML>
…
…
</HTML>
```

Notes:

- The tag [[#Split_Tag:: … #]] comes **before** the text that would be written to that split file.
- If the target folder of the split files does not exist, Visual CUT creates it on the fly.

Merging Text Files

Using a command line argument, you can instruct Visual CUT to merge any number of text files. For example, this feature allows you to append text or csv exports to an existing files. Another typical use scenario is to append a csv export (which doesn't include column headers) to a single-line text file that provides the column headers.
The command line argument structure is as follows:

… **"TXT_MERGE:Text_File_List>Text_File_Target>Top_Rows_To_Remove"**

The parameters (after the ":") are separated by a ">" and are as follows:

1. **Text_File_List:** comma separated list of the source files in the order they should be merged. If all source files share the same folder, **you can specify the full path just for the first file**. If a source file is not found, a warning is written to Failure.log and that file is skipped.

2. **Text_File_Target:** the file path & name for the resulting merged file. If the path is the same as the first source file, you may specify just the file name.

3. **Top_Rows_To_Remove:** the number of top rows to remove from each file being merged into the first file. This is useful if the first file already provides column headers that should be removed from the following files. Use '0' to indicate no rows should be removed.

For example:
"TXT_MERGE:c:\temp\File1.txt,File2.csv,File3.csv>c:\temp\Result.csv>0"

Dynamic Field Names
You can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. The dynamic content of these fields/formulas would be substituted into the command line argument.

Replacing Content in Text Files

Using a command line argument, you can instruct Visual CUT to replace any number of strings in a text file with specified substitutions. For example, this feature allows you to remove a Carriage Return and Line Feed at the end of csv file exports.
The command line argument structure is as follows:

… **"TXT_Replace:InFile||OutFile||find1>>replace1::find2>>replace2||Options"**

The parameters (after the ":") are separated by a "||" and are as follows:

1. **InFile:** the file path & name for the source text file.

2. **OutFile:** the file path & name for the resulting text file.
   - If no OutFile is specified, the source file gets updated
   - if the target folder doesn't exist, Visual CUT creates it
   - if no path is specified (just name) the path of the source file is used

3. **Find & Replace Pairs:** unlimited number of find and replace elements.
Each "find" element is separated from its "replace" element by a '**>>**'.
Each pair is separated from the next pair by a '**::**'
   - To specify special string characters, such as Carriage Return or Line Feed, use Chr() expressions (e.g., **Chr(10)** or **Chr(13)**.
   - To remove strings, specify them in the 'find' and leave the 'replace' as blank

4. **Options:** Leave blank, unless you wish to use a special option.
   - Specify END if you wish the substitution to occur only at the end of the file.

For example, to remove Carriage Return and Line Feed at the end of a text file, use this command line argument:

… **"TXT_Replace:c:\temp\Input.csv||Result.csv||Chr(10)>>::Chr(13)>>||End"**

Dynamic Field Names
You can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. The dynamic content of these fields/formulas would be substituted into the command line argument.

Removing Blank or Short Lines in Text Files

Using a command line argument, you can instruct Visual CUT to remove blank or short lines in text files. This is typically useful when one of the text export formats generates a file with unwanted blank lines or lines with just delimiters.
The command line argument structure is as follows:

… **"TXT_Remove_Short_Lines:InFile||OutFile||Max_Length_To_Remove"**

The parameters (after the ":") are separated by a "||" and are as follows:

1. **InFile:** the file path & name for the source text file.

2. **OutFile:** the file path & name for the resulting text file.
   - If no OutFile is specified, the source file gets updated
   - if the target folder doesn't exist, Visual CUT creates it
   - if no path is specified (just name) the path of the source file is used

3. **Max_Length_To_Remove:** Lines with that specified length or shorter would be removed.

For example, to remove all lines that are only 3 characters or shorter:

… **"TXT_Remove_Short_Lines:c:\temp\Input.csv||Result.csv||3"**

Dynamic Field Names
You can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. The dynamic content of these fields/formulas would be substituted into the command line argument.

Removing GUIDs from png Files Referenced in HTML Exports

This approach is designed to address a use scenario where each time you export a report to HTML, image files are created (charts, logos, etc.) with different names. The references to these files use a GUID to make the file name unique, like this:
<img src="Sales{EF6A442D-0DDA-436A-A178-2410853A7E05}.png"
Where the Sales portion is based on the export file name.

In most cases, the image files actually don't need unique GUIDy names and we can DeGUID (remove the GUID portion) from the file reference and from the actual png file name.

The DeGUID process detects if each png file is unique or a clone of another image within the same report (typical in cases of logos repeating on each page). If the png is not a clone (typical in reports with multiple charts), the GUID is replaced by an incrementing number (Sales_1.png, Sales_2.png, etc.). This ensures the local and web folders don't accumulate unique GUIDy names each time the scheduled process runs. It also saves space because while Crystal exports the same logo on different pages as multiple png files, the DeGUID process in Visual CUT automatically consolidates these clones as a single file.

You can elect to either delete the GUIDy png files (typical in cases the images are static (logos). Or you can elect to rename the png files to match the renamed references in the HTML source.

The command line argument structure is as follows (all in 1 line):
"TXT_DeGUID_png:InFile||Options"

The parameters (after the ":") are separated by a "||" and are as follows:

1. **InFile:** the file path & name for the source HTML file.

2. **Options:**
   DeleteLinkedFile to delete the referenced png files (Scenario 1 above)
   RenameLinkedFile to rename (DeGUID) the referenced png files (Scenario 2 above).
   Otherwise, **None**

For example, the following directive:
"TXT_DeGUID_png:c:\temp\Sales.htm||RenameLinkedFile"
Would remove all GUIDs from png file references in the HTML export. It will also rename the referenced png files by removing the GUIDs from their file names.

More complex scenarios (you wish to rename references to a logo image to a static name that doesn't depend on the export file name) can be handled by the more complex command line argument described in the next section (TXT_Replace_Tokens).

Dynamic Field Names
You can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. The dynamic content of these fields/formulas would be substituted into the command line argument.

Replacing Content in Text/HTML Files – Token Approach

This approach allows you to change the content of exported TEXT or HTML file.

The command line argument structure is as follows (all in 1 line):
"TXT_Replace_Tokens:InFile||OutFile||Start1^^End1^^replace1^^Location1^^Options1 ::Start2^^End2^^replace2^^Location2^^Options2||Global_Options"

The parameters (after the ":") are separated by a "||" and are as follows:

1. **InFile:** the file path & name for the source text file.

2. **OutFile:** the file path & name for the resulting text file.
   - If no OutFile is specified, the source file gets updated
   - if the target folder doesn't exist, Visual CUT creates it
   - if no path is specified (just name) the path of the source file is used

3. **Find & Replace Directives:** find and replace directives (separated from each other by "**::**").
Each directive has 5 elements separated by a '**^^**':
- **Start Token** (in the example above, this would be **`<img src="`**
- **End Token** (in the example above, this would be **`"`**
- **Replace with**: text to replace the text found between start and end tokens
- **Location**: **LAST** for last found match, **ALL** for all, **1** for 1st found match, **2** for 2nd …
- **Options**: **RenameLinkedFile** if you want to rename a matching file.
  DeleteLinkedFile to delete a matching file (if clean file name already exists).
  Otherwise, **None**

4. **Global Options:** Leave blank

For example, the following directive:

"TXT_Replace_Tokens:c:\temp\Sales.htm||||<img src="^^" border="0" **width="52px" height="32px"**> ^^LionHead.png^^ALL^^RenameLinkedFile||"

you can replace multiple image files, typically named as variations of:
Output_File{1A49A55D-544F-472F-B40F-5B7062C3D47A}.png with a single image file.

"TXT_Replace_Tokens:c:\temp\Sales.htm||c:\temp\Sales.htm||<img src="^^ " border="0" **width="52px" height="32px"**> ^^**MyLogo.png^^ALL^^RenameLinkedFile**||"

will change all references to an image with dimensions of 52x32 with a single static reference to MyLogo.png. Due to the **RenameLinkedFile** directive all the temporary matching files such as **Output_File{1A49A55D-544F-472F-B40F-5B7062C3D47A}.png**
Will be renamed to **MyLogo.png**
This has 2 effects:

1. Less clutter in the export folder because multiple image files may be replaced by a single image file
2. Less files need to be attached to an email message or uploaded to a web server
3. The web server and the client browser can cache MyLogo.png, improving performance

Dynamic Field Names
You can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. The dynamic content of these fields/formulas would be substituted into the command line argument.

Inserting Base64-Encoded Files Inside Text/XML

This option was developed for a customer who needed to export invoice data to an XML file containing an embedded image encoded with Base64 (a method that converts from binary to text representation). The XML file is then transmitted to a business partner using SFTP_Upload.

The process takes the following steps:

1. A TEXT export creates the XML file with a reference to a file that should be embedded at a particular location. The reference has the following structure:
   [Insert_File_Base64:file_path_and_name]]
   for example,
   [Insert_File_Base64:C:\Temp\Invoice_32556_image.pdf]]

2. Using a **TXT_Replace_Base64** command line argument directs Visual CUT to search the XML file, locate such references, and replace them (including the surrounding tags) with the Base64 encoding of the specified file.

The command line argument structure is as follows (all in 1 line):
"TXT_Replace_Base64:InFile||OutFile||Options"

The parameters (after the ":") are separated by a "||" and are as follows:

1. **InFile:** the file path & name for the source text file.

2. **OutFile:** the file path & name for the resulting text file.
   - If no OutFile is specified, the source file gets updated
   - if the target folder doesn't exist, Visual CUT creates it
   - if no path is specified (just name) the path of the source file is used

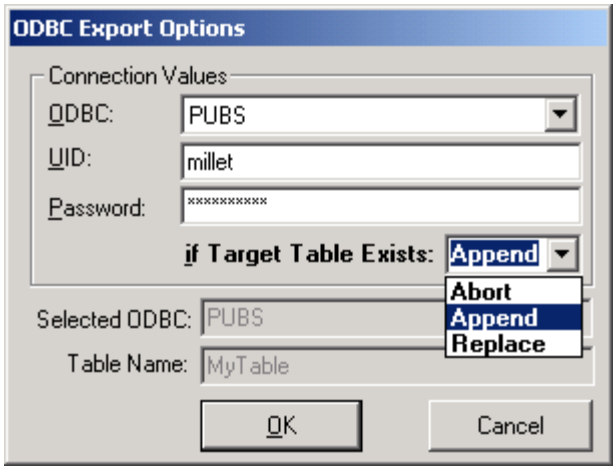
3. **Options:** Leave blank

Notes:

- You can have any number of references embedded in the exported text file. Visual CUT will replace all of them.
- You can use field or formula names within the command line argument (just like you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT. The dynamic content of these fields/formulas would be substituted into the command line argument.

ODBC Export Functionality

Visual CUT (for Crystal 9 and above) provides ODBC Export options that are not available in Crystal alone. As shown below, the ODBC Export Options dialog allows you to select **Abort** (default Crystal action), **Append**, or **Replace** as the action when the target table already exists:



Note: the **Table Name** is controlled by the **Export File Name** (that option is not on the dialog shown here but on the main email/export processing options tab of Visual CUT). Since that option accommodates dynamic content, you can control the table name using fields/formulas in the report…

Append Functionality

By selecting append, you can add data to an existing table. One use for this is for **recording snapshots of data over multiple periods of time**. For example, your database may show current inventory levels or current account balances but it is very difficult to generate a report that shows inventory levels or account balances at the end of each month for the last few years. By scheduling Visual CUT to run at the end of each month and append information reflecting the inventory levels or the account balances at that time, you enable easy tracking of that information over time. In Data Warehousing jargon, this is called a snapshot data warehouse.

Another use scenario is to record the fact that certain records were processed by Visual CUT and to avoid duplicate emails. For example, as orders arrived throughout the day, you can schedule Visual CUT to run every 5 minutes and email order confirmation messages to the customers. By appending to an ODBC table, you can record which orders have already been confirmed. Alternatively, you can use the **Skip_Recent** command line argument described in the section Avoiding Duplicate Processing on page: 52.

Another use scenario is to **Extract, Transform, and Load (ETL) data into a data warehouse or across data sources** without needing expensive ETL tools or consultants.

Replace Functionality

A typical use for the Replace option is to **generate temporary tables for use by other processes or crystal reports**. For example, Crystal Reports can't aggregate other aggregate values (e.g., average group totals) and can't sort groups based on formulas that use aggregate values. However, you can create a batch file that calls Visual CUT twice: first to export a report with aggregate values to a temporary table and then to process a 2nd report that uses the temporary table as input.

Capturing & Processing Incoming Emails

Visual CUT can automatically capture incoming emails from your email server to a database table. A Crystal report using the data in that table can then be processed in Visual CUT to trigger reactions to that email, such as:

- Export/Print/FTP/Email information or reports
- Update databases using **After_Success_SQL**
- Trigger other applications or Visual CUT processes using **After_Burst_Batch**

Use Scenarios

Here are a few examples of the types of workflow automation this enables:

- **Requesting a Report via a Simple Web Form or via Email:**
  - A manager, employee, customer, or supplier can use a simple web form (**no web application required**) to request a report.
  - The information gets emailed to your email server
  - Visual CUT captures the email into the EMAIL_CAPTURE table.
  - A Crystal report using that table inside Visual CUT uses **After_Burst_Batch** to trigger another Visual CUT process by specifying a report to run, parameters, email destination, etc.
  - The triggered Visual CUT report runs and emails the output to the requesting employee/customer.

  Note: using the same exact process, you can let customers request information or reports by sending an email. For example, a customer may request information about the status of their order, the balance in their account, etc.

  Here is a sample email from ABP Reports' Boomerang system, which uses this technique, allowing users to request reports via email by simply clicking on a hyperlink:

Thank you for using the Boomerang automated reporting service.

You have recently requested a listing of all your available reports by sending a HELP email.

Below are all the reports you currently have access to request via email.
To request a report simply copy the subject text in a new email and add it to your Subject line, or simply click on the link icon.

All email requests should be sent to boomerang@yourcompany.com

Available reports: 31-Jul-2013

Report Id	Report Name	Emails sent	Last sent	Email Subject	Link
12	Pending Invoices	6	31-07-2013	[BOOMERANG::12]	✉
20	Current Status	4	31-07-2013	[BOOMERANG::20]	✉

- **Updating a Database via a Simple Web Form:**
  - A manager, employee, customer, or supplier can provide information (for example, complete a registration for an event or provide order status information) using a simple web form (**no web application required**).
  - A Submit button triggers an email with the form information to your email server.
  - Visual CUT captures the email into the EMAIL_CAPTURE table
  - A Crystal report using that table inside Visual CUT uses **After_Burst_SQL** to update another database table (for example, inserting the information into a REGISTRATION table.

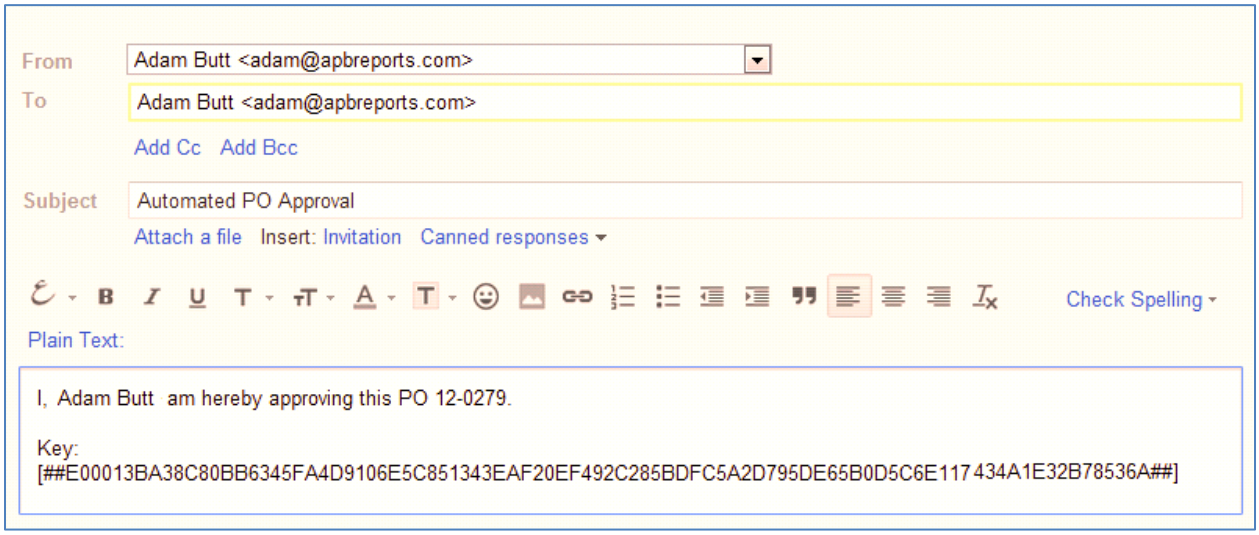
- **Collecting Customer Feedback & Updating a Database via Email Links:**
  - You can use Visual CUT to burst Customer Satisfaction survey emails when a purchase, course, visit, RFQ, event, or tech support case is closed.
  - Inside the HTML email message you embed 5 mailto hyperlinks corresponding to ratings of: **Poor**, **Fair**, **Good**, **Very Good**, and **Excellent**.
  - This allows the customer to simply click on one of these links to trigger an email back to you with the appropriate ratings.
  - Visual CUT captures the email into the EMAIL_CAPTURE table.
  - A Crystal report using that table inside Visual CUT uses **After_Burst_SQL** to insert feedback records into a CUSTOMER_FEEDBACK table.
- **Importing Data from Email Attachment into a Database Table:**
  - Email with excel file attachment is captured and Visual CUT extracts the attachment to a specified folder.
  - 1st Crystal report reads records from the **Email_Capture** table and bursts to a dummy VC_Skip_Export.
  - **After_Burst_Batch** triggers a 2nd report that uses Excel as data source, but redirects it (using **Database_Path** command line argument) to retrieve data from the email attachment file.
  - Visual CUT exports the data from the Excel file via ODBC and appends it to a specified table.

- **Requesting & Capturing Management Decisions Via Email:**
  - Imagine you have a PO table with Purchase Order records that are first inserted with a status of 'Requested'.
  - You can use Visual CUT to burst and email to the manager in charge of each department Approval Requests for these POs.
  - Attached to each emails you would provide a PDF file with detailed information about the requested Purchase Orders.

- Inside the HTML email message body, you embed 2 mailto hyperlinks allowing the manager to trigger a **Reject** or **Approve** email back to you.
  Note: APB Reports (a BI Consulting firm with impressive record of leveraging Visual CUT for a variety of use scenarios) implemented such a process for one of their clients. Here is what the email message looks like:

Automated PO Approval: 12-0543 (04-07-2012)
The attached Purchase Order requires your approval:
Rig: N09
PO No.: 12-0543
PO Total USD: 239580 USD
Description: R&R - TO SEND RISERS TO ALPHATEC FOR REPAIRING
Approval options. Click on one of the links below:
Approve
Reject
Service provided by APB Reports
8863-1003000052903-1003000110947-04-07-2012.pdf
69K View Download

- Here is the email message resulting from clicking on the Approve hyperlink:



- Note that the email message contains an encrypted text. This ensures the integrity of the emailed information can't be compromised on its way back to us.
- The encrypted text contains all the necessary data to identify the PO and indicate rejection or approval (each hyperlink has a different decision code embedded in the encrypted text).

- The encrypted text for each **emailto** hyperlink is generated via a Crystal formula using the **BlowFishEncrypt()** function provided by the CUT Light UFL.
- The email triggered by clicking on the Reject or Approve hyperlink gets sent to your email server
- Visual CUT captures the email into the EMAIL_CAPTURE table.
- A Crystal report using that table inside Visual CUT decrypts the text in the email message body (or subject line) using the **BlowFishDecrypt()** function provided by the CUT Light UFL.
- Using **After_Burst_SQL**, Visual CUT then updates the status of the Purchase Order to 'Approved' or 'Rejected'. The SQL statement may look something like this:

```
'UPDATE PO SET PO_Status = '& {@NEWSTATUS} & ','& (IF {@NEWSTATUS}
= '8' THEN (' PO_COMMENT = '& "'PO approved by "& {@EMAIL} & """)ELSE('
PO _COMMENT = '& "'PO rejected by "& {@EMAIL} & """))&' WHERE PO_ID = '
& {@PO_ID} & ' AND  PO_STATUS = 6'
```

Triggering Email Capture

An Email Capture process gets triggered by running a report in Visual CUT using a command line with an "**Email_Get:Directives_Key**"argument. For example (all in 1 line):

```
"C:\Program Files\Visual CUT 11\Visual CUT.exe" -e "C:\Test\Email_Capture.rpt" "Email_Get:P2"
```

The Directive key is used to look up in the **DataLink_Viewer.ini** a matching key under the [Email_Get] section.

[Email_Get] ini Section

Each Directives Key can specify multiple directives separated by a '||' delimiter:

```
[Email_Get]
P1=Get_PO_Decisions
P2=Get_Evaluation_Requests||Get_Camp_Feedback||Get_Booking_Requests
```

Notes:
1. The Email_Get process gets triggered BEFORE the report actually retrieves data from the database.
2. The ODBC DSN used by the report is automatically used as the ODBC DSN target for loading captured emails into the target table. Visual CUT reuses the saved login information for the report in order to update the target email capture table.
3. The target table is specified in the directive ini section (see next page).
4. The tables used by the report are not restricted to the email capture target table.
5. By default, processing of the report is aborted if zero new emails where captured and inserted into the database. To change that behavior, set Abort_Report_Process_If_No_New_Emails in the directive ini section to False.

Email Get Directive Sections

Each email capture directive must have its own ini section as follows:

```
[Get_Evaluation_Requests]
//   Required Entry.  Not case sensitive.  Example: Filter=To = "ido@MilletSoftware.com"
//   Filter=(Subject contains "test" AND From like "*@MIlletSoftware.com*") OR (body Contains "test")
Filter=(Subject contains "evaluation request")
// optional. Default is 50 body lines downloaded for filtering. Reduce if no body filter. Increase if HTML message.
Body_Lines=1
//   Required Entry.  with 1440, only messages sent in the last 24 hours are captured
Message_Sent_in_Last_N_Minutes=1440
Server=mail.milletsoftware.com
// set PopSSL to True if TLS/SSL (encrypted communication) is used when getting emails from the server
PopSSL=False
// set Pop3STLS to True (and PopSSl to False) if unencrypted connection (typically port 110)
//        automatically converts  to a secure TLS connection via the STLS command.
Pop3STLS=False
// Set Pop3SPA to True to use SPA authentication
Pop3SPA=False
Port=110
//   If email server User id is not specified, it is assumed to be same as SMTP setting
User_ID=ido@MilletSoftware.com
//   copy from [Options] section or leave blank to use same. If you need a different password, temporarily set the
//     default SMTP password to the desired password. Save. Copy the encrypted Password, and then reverse.
Email_Password_Encrypted=ECDC1AABC705D61F04F6A16F61
//   should captured emails (if successfully inserted into the table), be deleted from the email server?
Delete_Email_From_Server=True
//   if specified, captured emails get saved as eml files to this folder (Safety Measure if you delete captured emails)
Save_As_EML_To_Folder="C:\VC\Captured_Emails\"
//   if specified, attachments are saved as files to this folder. By default, if the file already exists, it gets overwritten.
Save_Attachments_To_Folder="C:\VC\Captured_Email_Attachments\"
//   optional. Default is False. If set to True, instead of overwriting, 4 characters are added to make the name unique
//            note: if set to True AttachmentList info captured to the database table would reflect the unique names
Save_Attachments_To_Unique_File_Names=True
//   Table where email records are to be inserted. NOTE: If an existing record is found from
//       same Sender, same Subject, and same sending DateTime, the record doesn't get inserted a second time.
Table=Email_Capture
Last_Success=12:14:56-12:15:13=00:00:17 InBox:11 / Targeted:8 / Inserted:6
```

Capturing Emails

The process targets email messages that are stored at the specified email server for the specified User ID. Obviously, each capture process may want to target only a subset of these messages. To keep the process as efficient as possible, the capture progresses through two main phases:

- Header Download & Filtering
- Targeted Download & Database Capture

The following sections provide more detail about these phases.

Phase 1: Header Download & Filtering

Visual CUT first downloads just the email headers (including first 50 lines of the message body) for all messages stored on the mail server for the specified User ID. While this process avoids the need to also download attachments, if you wish to keep the process fast, don't let the Inbox for that User grow to too many messages. You can manually delete old email messages in the server Inbox,. You can also set **Delete_Email_From_Server** to True, to automatically delete emails from the server after Visual CUT inserts them into the specified database table.

For backup purposes, particularly if you are just starting to use this process, if you elect to delete emails from the server, you should set the **Save_As_EML_To_Folder** option. Visual CUT would then deposit all processed emails (those that survived the **Filter** and **Message_Sent_in_Last_N_Minutes** conditions) as eml files. These files can be opened in the free Windows Mail or Outlook Express. All file attachments are embedded inside the eml files, so you can be secure in knowing you have an archive of the whole message.

The **Filter** and **Message_Sent_in_Last_N_Minutes** take the initial set of partial downloads and produce a targeted subset of email messages that are then downloaded with full body as well as attachments.

The **Filter** condition allows you to apply simple or composite conditions to any email property such as From, To, Subject, and Body. Here are a few examples:

```
Filter=Body like "Report Request*"
Filter=(Subject contains "PO Approved" OR Subject contains "PO Rejected") AND
           From contains "@MilletSoftware.com"
```

The expressions are **not case sensitive**.

Supported operators: **CONTAINS**, **LIKE**, **=**, **< ,>**, **<=**, **>=**, **<>**

The "*****" wildcard matches 0 or more occurrences of any character.

```
Parentheses can be used to control the logic.
```

Phase 2: Targeted Download & Database Capture

After establishing a subset of targeted emails, Visual CUT downloads the full content of these emails. For each of these fully downloaded emails, Visual CUT takes the following steps:

1. If **Save_As_EML_To_Folder** is specified, the full email message (with embedded attachments) is archived to the specified folder
2. A connection is established to the database server used by the report via the ODBC DSN used by the report and the encrypted login information stored for the report.
3. The existence of the specified table for capturing email information is confirmed
4. If the same email message (same subject, send DateTime, and From info) doesn't already exist in the table, the email information is inserted into the table.
5. If the message got inserted into the table, and you set the **Delete_Email_From_Server** option to True, Visual CUT asks the email server to delete the message.

Monitoring Results of the Process

If a failure occurs during the process, the usual failure alerts and logging processes apply.
If no failure occurs, Visual CUT updates the ini section for the processed directive with an entry such as this:

Last_Success=**12:14:56-12:15:13=00:00:17** InBox:**11** / Targeted:**8** / Inserted:**6**

This allows you to see:

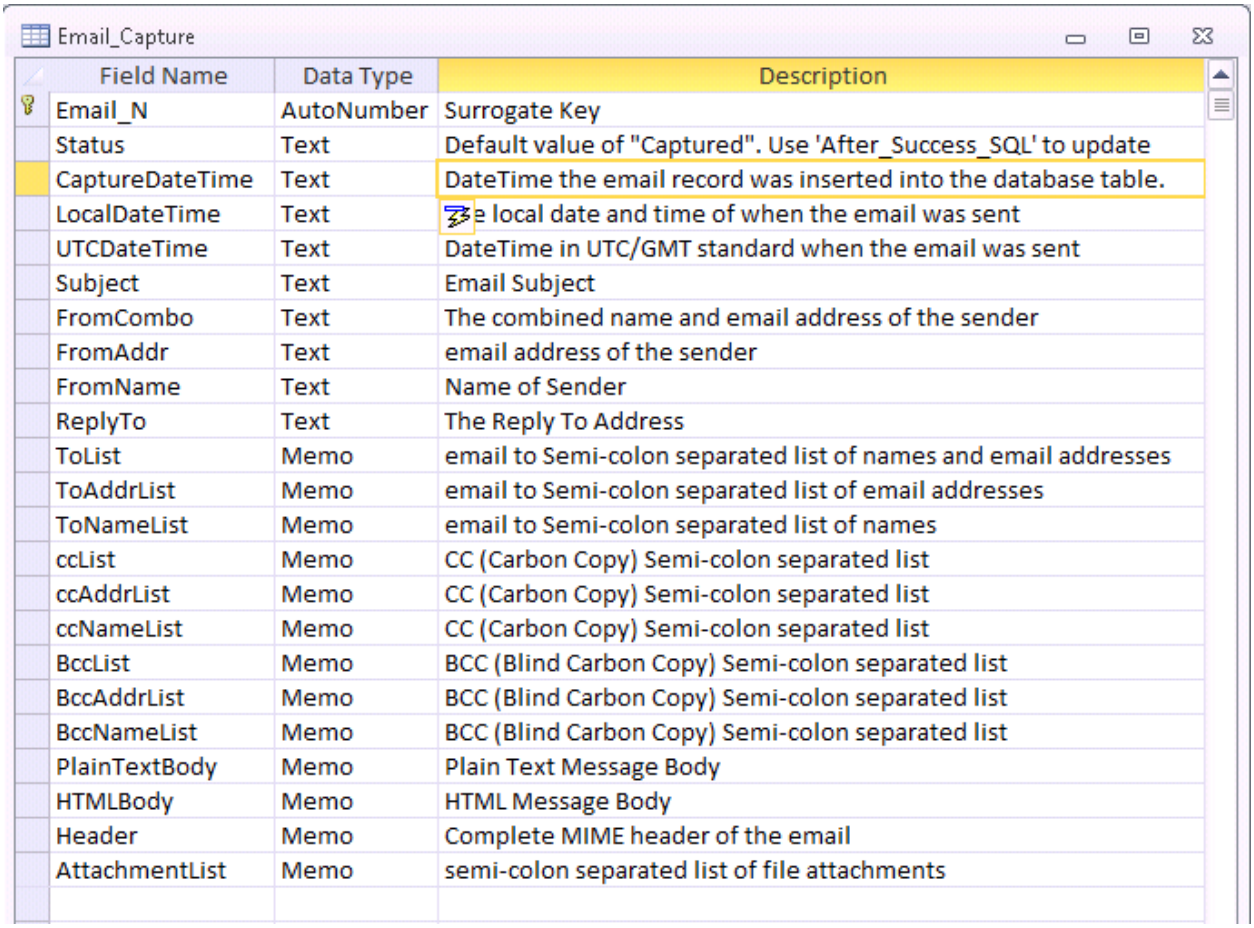
- **Start Time**, **End Time**, and **Total Time**,
- Number of Messages In the **InBox**
- Number of Messages **Targeted** after applying Filter & **Message_Sent_in_Last_N_Minutes**
- Number of Messages **Inserted** to the Table (after skipping existing records)

Email Capture Table Structure

You can name a different table for each email capture directive. But an email capture table must have the same data structure.

MS Access Table Structure

Here is the table structure for MS Access:



Email_Capture

Field Name	Data Type	Description
Email_N	AutoNumber	Surrogate Key
Status	Text	Default value of "Captured". Use 'After_Success_SQL' to update
CaptureDateTime	Text	DateTime the email record was inserted into the database table.
LocalDateTime	Text	e local date and time of when the email was sent
UTCDateTime	Text	DateTime in UTC/GMT standard when the email was sent
Subject	Text	Email Subject
FromCombo	Text	The combined name and email address of the sender
FromAddr	Text	email address of the sender
FromName	Text	Name of Sender
ReplyTo	Text	The Reply To Address
ToList	Memo	email to Semi-colon separated list of names and email addresses
ToAddrList	Memo	email to Semi-colon separated list of email addresses
ToNameList	Memo	email to Semi-colon separated list of names
ccList	Memo	CC (Carbon Copy) Semi-colon separated list
ccAddrList	Memo	CC (Carbon Copy) Semi-colon separated list
ccNameList	Memo	CC (Carbon Copy) Semi-colon separated list
BccList	Memo	BCC (Blind Carbon Copy) Semi-colon separated list
BccAddrList	Memo	BCC (Blind Carbon Copy) Semi-colon separated list
BccNameList	Memo	BCC (Blind Carbon Copy) Semi-colon separated list
PlainTextBody	Memo	Plain Text Message Body
HTMLBody	Memo	HTML Message Body
Header	Memo	Complete MIME header of the email
AttachmentList	Memo	semi-colon separated list of file attachments

This table is included in the **Visual_CUT_Log_Sample.accdb**
If you can't find it in the Visual CUT application folder, email me and and I'll send you a copy.

SQL Server Table Structure

Here is a script (provided by APB Reports) for creating the table structure in SQL Server:

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_PADDING ON
GO
IF (SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES
      WHERE TABLE_NAME = 'APB_DW_EMAIL_CAPTURE') IS NULL
BEGIN
CREATE TABLE APB_DW_EMAIL_CAPTURE
(
[Email_N] [NUMERIC](15,0) IDENTITY(1,1)PRIMARY KEY CLUSTERED,
[Status] [VARCHAR](100) DEFAULT 'CAPTURED',
[CaptureDateTime] [VARCHAR](100) NULL,
[LocalDateTime] [VARCHAR](100) NULL,
[UTCDateTime] [VARCHAR](100) NULL,
[Subject] [VARCHAR](500) NULL,
[FromCombo] [VARCHAR](200) NULL,
[FromAddr] [VARCHAR](200) NULL,
[FromName] [VARCHAR](200) NULL,
[ReplyTo] [VARCHAR](200) NULL,
[ToList] [text] NULL,
[ToAddrList] [text] NULL,
[ToNameList] [text] NULL,
[ccList] [text] NULL,
[ccAddrList] [text] NULL,
[ccNameList] [text] NULL,
[BccList] [text] NULL,
[BccAddrList] [text] NULL,
[BccNameList] [text] NULL,
[PlainTextBody] [text] NULL,
[HTMLBody] [text] NULL,
[Header] [text] NULL,
[AttachmentList] [text] NULL);
END
GO
```

Oracle Table Structure

Here is a script (provided by APB Reports) for creating the table structure in Oracle:

```
CREATE TABLE EMAIL_FEEDBACK_CAPTURE
(
"EMAIL_N" NUMBER NOT NULL,
"STATUS" VARCHAR2 (100) DEFAULT 'CAPTURED',
"CAPTUREDATETIME" VARCHAR2 (100) NULL,
"LOCALDATETIME" VARCHAR2 (100) NULL,
"UTCDATETIME" VARCHAR2 (100) NULL,
"SUBJECT" VARCHAR2 (500) NULL,
"FROMCOMBO" VARCHAR2 (200) NULL,
"FROMADDR" VARCHAR2 (200) NULL,
"FROMNAME" VARCHAR2 (200) NULL,
"REPLYTO" VARCHAR2 (200) NULL,
"TOLIST" CLOB NULL,
"TOADDRLIST" CLOB NULL,
"TONAMELIST" CLOB NULL,
"CCLIST" CLOB NULL,
"CCADDRLIST" CLOB NULL,
"CCNAMELIST" CLOB NULL,
"BCCLIST" CLOB NULL,
"BCCADDRLIST" CLOB NULL,
"BCCNAMELIST" CLOB NULL,
"PLAINTEXTBODY" CLOB NULL,
"HTMLBODY" CLOB NULL,
"HEADER" CLOB NULL,
"ATTACHMENTLIST" CLOB NULL,
"REC_DELETED" NUMBER DEFAULT 0,
CONSTRAINT APB_DW_EMAIL_CAPTURE_PK PRIMARY KEY (EMAIL_N)
USING INDEX TABLESPACE IPSINDEX
)
TABLESPACE IPSDATA;

CREATE SEQUENCE EMAIL_N_SEQ START WITH 1 INCREMENT BY 1;

CREATE OR REPLACE TRIGGER EMAIL_CAPTURE_INSERT
BEFORE INSERT ON EMAIL_FEEDBACK_CAPTURE
FOR EACH ROW
BEGIN
   SELECT EMAIL_N_SEQ.NEXTVAL INTO :NEW.EMAIL_N FROM DUAL;
END;
/

COMMIT;
```

Monitoring Visual CUT Processing

Visual CUT provides several ways to monitor, log, and alert users about processing failures:

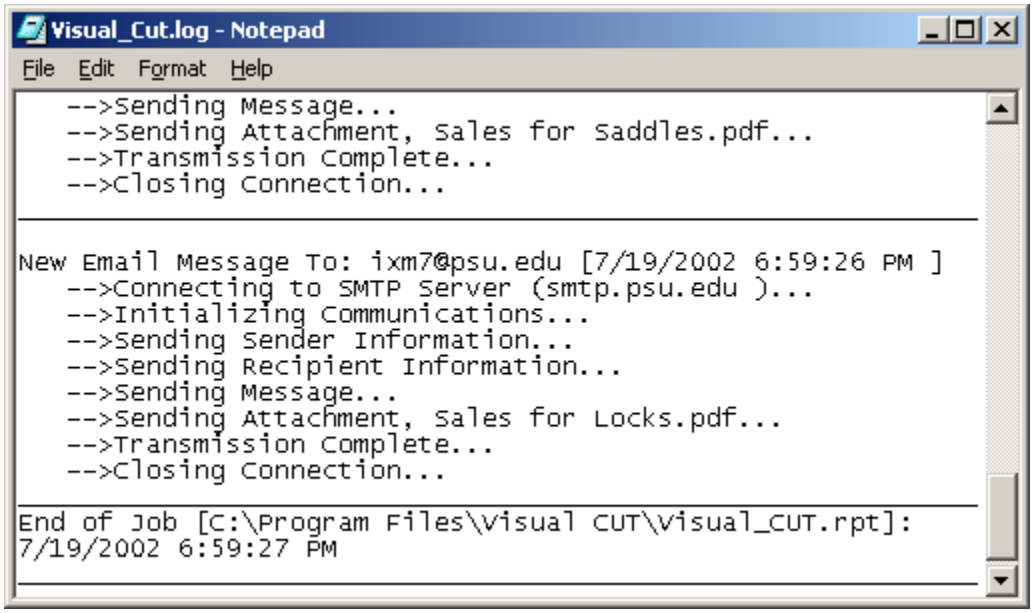
- An **email alert** can be sent to a specified address when a failure occurs.
- Processing can be recorded in an **ODBC database**
- Failures can be recorded in a **Failure.log text file** or presented in **Message boxes**
- **Email communications** with the SMTP server can be logged to Visual_Cut.log
- Job status information can be monitored by other applications by asking Visual CUT to signal successful or failed processing via **text job status files**.

This section discusses how you can use these options.

Logging and Monitoring Visual CUT

'Log Email Activity' option

This option, located at the bottom-right corner results in logging of all e-mail activity, including failure and success outcomes, to **Visual_Cut.log**
This text file will be placed in the same directory where Visual CUT was installed (typically C:\Program Files\Visual Cut\). It can be opened by *Notepad* or any word processor and is automatically created by enabling this option. Here is what this log file looks like:



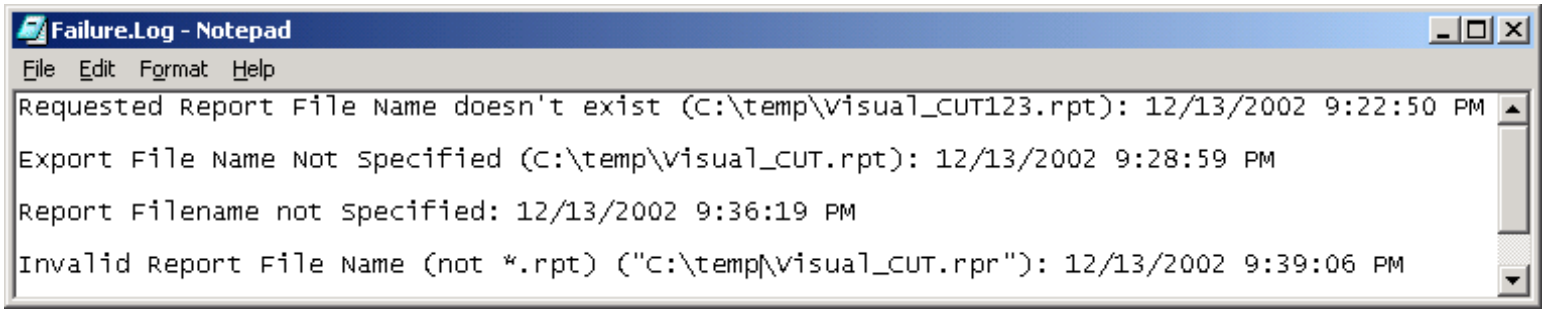
Silent Unattended Failure Option

When Visual CUT processes a report it may encounter various problems such as missing destination e-mail address, non-existing or invalid report file names, and missing export file names. Such cases halt the execution and trigger an appropriate message box.

If you want to ensure that Scheduled/Unattended processing doesn't stop when encountering such cases, open **DataLink Viewer.ini** (located where you installed Visual CUT and automatically created the 1st time you run Visual CUT) and specify the following option:

[Options]
Silent_Unattended_Failure=TRUE

This allows silent logging and skipping of any failure and avoiding message boxes. The details about each failure get logged into a text file (***Failure.Log***) located where you installed Visual CUT. Here is an example of failure messages in this file:


Failure.Log - Notepad
File Edit Format Help
Requested Report File Name doesn't exist (C:\temp\Visual_CUT123.rpt): 12/13/2002 9:22:50 PM
Export File Name Not Specified (C:\temp\Visual_CUT.rpt): 12/13/2002 9:28:59 PM
Report Filename not Specified: 12/13/2002 9:36:19 PM
Invalid Report File Name (not *.rpt) ("C:\temp\Visual_CUT.rpr"): 12/13/2002 9:39:06 PM

Note: If you turn on this option and you have a multi-line batch file that invokes processing of multiple reports, you may want, within your batch file, to have branching logic based on the success or failure of each line. In order to respond (for example via Batch File IF THEN logic) to Success or Failure of Visual CUT processing, you can check for the existence of **Success.txt** (in the same directory where Visual CUT is installed) after each invocation of Visual CUT processing via a command line.

You turn on this option by opening **DataLink Viewer.ini** (located where you installed Visual CUT and automatically created the 1st time you run Visual CUT) and specifying the following option:

[Options]
Create_Success_File=TRUE

Silent Attended Failure Option

This option is identical to the Silent Unattended Failure option discussed in the previous section. The difference is that it applies to cases where Visual CUT processing is invoked interactively (by pressing the START button). This is particularly useful in cases when the processing takes a long time and the user doesn't want to be tied to the screen.

When Visual CUT processes a report it may encounter various problems such as missing or invalid destination e-mail address, non-existing or invalid report file names, and missing export file names. Such cases halt the execution and trigger an appropriate message box.

If you want to ensure that Attended (interactive) processing doesn't stop when encountering such cases, open **DataLink Viewer.ini** (located where you installed Visual CUT and automatically created the 1st time you run Visual CUT) and specify the following option:

[Options]
Silent_Attended_Failure=TRUE

When this option is turned on, any errors (e.g., one of the Groups didn't have an e-mail address associated with it) are recorded in ***Failure.log*** (a plain text file) instead of halting the execution with a message box. See previous section for an example of what this log file looks like.

Here is an example of such an error message in the Failure.log:

An attempt to send e-mail failed for the following reason(s): No Recipient E-mail Address Specified
C:\Program Files\Visual CUT\Visual_CUT_with_Deliberate_Email_Problems.rpt: 4/9/2003 1:06:43

Avoiding Duplicate Processing

Imagine you need to email "Order Received" (or "Order Shipped") confirmations to your customers. You use Visual CUT to schedule bursting of a report grouped by Order_ID, selecting all orders that were received in the current day. You wish to run the report every 30 minutes without repeating emails.

To solve this problem, you can use a command line argument that instructs Visual CUT to skip processing if the target export file already exists and was created less than N minutes ago.
For example, using the following scheduling string (or batch file):

"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Program Files\Visual CUT\Visual_CUT.rpt" **"Skip_Recent:2880"**

---

would cause Visual CUT to start processing the Visual CUT.rpt and burst the first group value ("Competition"). However, Visual CUT would skip this bursting step (and continue to the next group value) if the target export file exists and was created in the last 48 hours (2880 minutes).

In your 'Order Received' notification, you may not need to export and attach any file to the email notification. However, in order to use the Skip_Recent functionality, you would instruct Visual CUT to export in each bursting step to a file name containing the Order_ID. This file would then be used by the Skip_Recent logic, even though you will not attach the file to the email message.

Non-Bursting Scenario:
You can take advantage of this functionality even in cases where no bursting is taking place by having the exported file name reflect the count or maximum record number in a table. Using the Skip_Recent command line argument, you can then ensure that unless a new record has been added to the table (and hence the target export file name is new) Visual CUT processing would be aborted.

Alternative Approaches:
As alternatives to using ***Skip_Recent*** you can:

1. Turn on the option to log Visual CUT processing using the functionality described in: "**Record Processing to an ODBC Database**" and, in your report, outer join to the **MS_Log** table to check if the current group value has already been successfully processed.

2. Use ***After_Success_SQL*** to generate an UPDATE sql statement to set the value of a "Processed" or "Date Sent" column.

Avoiding Too Many Active Visual CUT Instances (Queuing)

Many Visual CUT users have increased their use of the tool to the point where several instances of Visual CUT may be actively processing reports at any given time. In most cases, you can reduce the number of concurrent instances by simply combining multiple command lines into a single batch file. In other cases, you can control the maximum number of active Visual CUT instances using the following DataLink_Viewer.ini entry:

[Options]

…

Maximum_Allowed_Active_Instances=4

When launching a new Visual CUT instance, if the number of active Visual CUT instances reached that maximum number, the new instance is placed into sleep. It "wakes up" every 5 seconds to check if it can launch. Once the number of active instances drops below the maximum allowed level, the new instance is allowed to launch.

Job Status Functionality

During unattended/scheduled processing, Visual CUT generates a **job status text file**, located in the Visual CUT application folder and named:
VC_Job_Status_N.txt (containing the error message) if a failure occurred
- or -
VC_Job_Status_Y.txt if processing was successful.
Visual CUT erases these files (if they exist) at the start of each unattended processing.

Other applications that trigger Visual CUT processing via a command line call can check for the existence of these job status files as an indicator for processing status. For example, a web application can use this option to invoke Visual CUT processing (e.g., an export to a PDF file with bookmarks) and then keep checking for one of the job status file names before continuing. If the success file (**VC_Job_Status_Y.txt**) is found, redirect the user's browser to the resulting PDF file – if the failure file is found (**VC_Job_Status_N.txt**), present the user with the error message inside that text file.

In order to support job status monitoring in cases where multiple instances of Visual CUT may be processing report requests at the same time, your application can specify a **unique job status file name for each call to Visual CUT** .
For example, if your command line invocation of Visual CUT is:
"C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Test\Report.rpt"
"JOB_STATUS_ID:4523"
then the resulting job status file for that call would be **4523_Y.txt** or **4523_N.txt**

A DataLink_Viewer.ini file option (**Job_Status_Path**) under a **[File_Locations]** section allows you to override the default location (Visual CUT application folder) for the Job Status indicator files. For example:

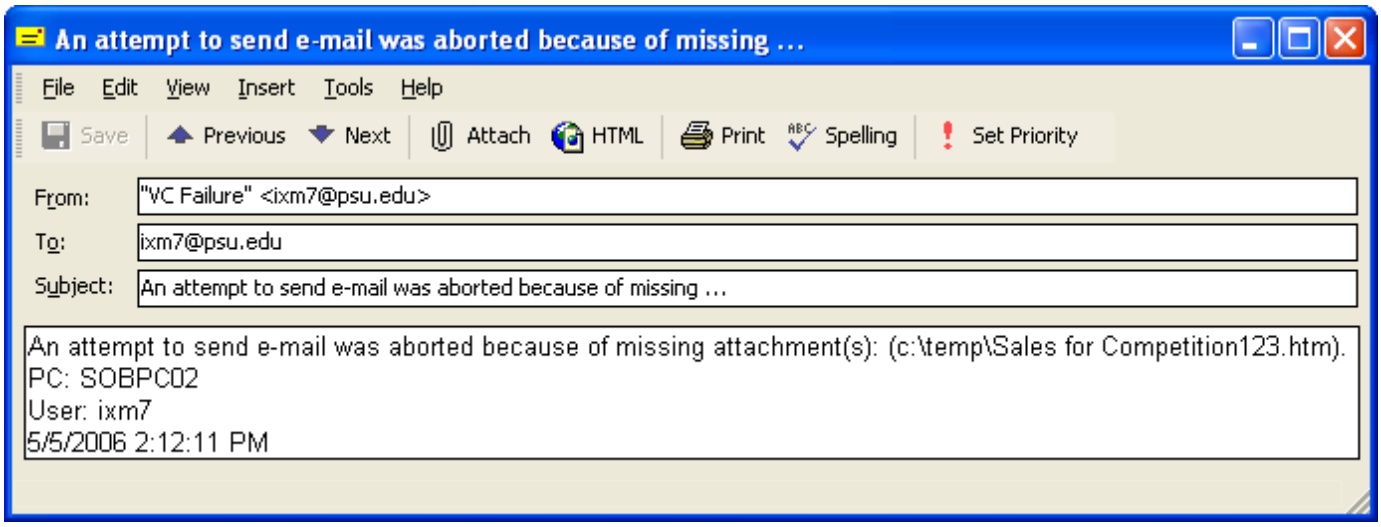
```
-----------------------------------
[File_Locations]
Job_Status_Path=C:\Temp\
-----------------------------------
```

This is useful in situations where a web application (PHP, ColdFusion, ASP.NET…) is not allowed read access to the Visual CUT application folder.

Note: this functionality is disabled if the Generate Success.txt file option is turned off.

Failure Alerts via Email

The Visual CUT Option dialog allows you to specify an email address that should receive a message whenever Visual CUT encounters a processing failure. Here is an example of such a message:


An attempt to send e-mail was aborted because of missing ...
File Edit View Insert Tools Help
Save Previous Next Attach HTML Print Spelling Set Priority
From: "VC Failure" <ixm7@psu.edu>
To: ixm7@psu.edu
Subject: An attempt to send e-mail was aborted because of missing ...
An attempt to send e-mail was aborted because of missing attachment(s): (c:\temp\Sales for Competition123.htm).
PC: SOBPC02
User: ixm7
5/5/2006 2:12:11 PM

Record Processing to an ODBC Database

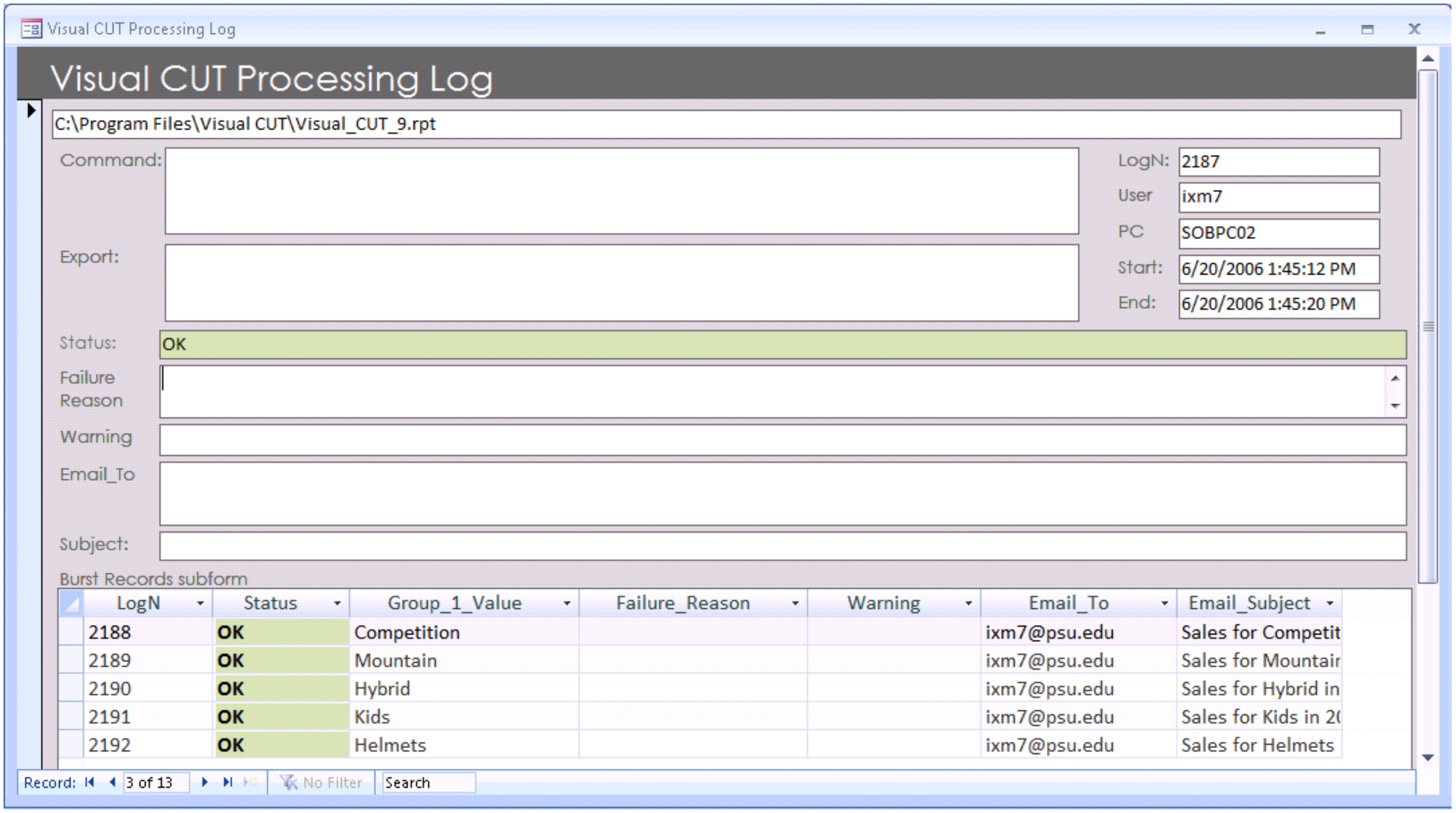
In order to log processing to an ODBC database, you must create a table called **MS_Log** in the target database. This can be in MS Access, SQL Server, Oracle, etc.

MS Access Database Sample

MS_Log : Table

Field Name	Data Type	Description
LogN	AutoNumber	Surrogate Key
Parent_LogN	Number	If bursting step, shows the LogN of the parent record. Contains 0 if this is the Parent Record
Rpt_Path_Name	Text	rpt file being processed
Proc_Start	Text	String representation of Prcessing Start DateTime (not using DateTime to allow use by any DBMSs)
Proc_End	Text	String representation of Processing End DateTime (not using DateTime to allow use by any DBMSs)
Group_1_Value	Text	The value of the current group. Blank if Process Level is 0
Status	Text	Started, Failed, OK
Failure_Reason	Memo	Why did processing fail?
Warning	Memo	Blank Unless a warning was recorded
Export_File_Name	Memo	Blank if EXPORT mode is WHOLE and this is a Bursting step (exporting recorded at Parent Level)
Email_To	Memo	Blank if EMAIL mode is WHOLE and and this is a Bursting step (emailing recorded at Parent Level)
Email_Subject	Text	
Command_Line	Memo	The command line text (beyond the exe file name). Blank if this is attended execution.
User_ID	Text	The Windows User ID running the process
PC_ID	Text	The Windows machine running the process (several machines can log to a single ODBC target)

Above is the table structure required for this table, when implemented under MS Access: A sample MS Access database is available for **download**. That sample database contains a form (with a subform) showing how you can view processing information, including bursting steps:



SQL Server Instructions

Here is a script for creating the table in SQL Server:

```
CREATE TABLE [dbo].[MS_Log] (
[LogN] [int] IDENTITY (1, 1) NOT FOR REPLICATION  NOT NULL ,
[Parent_LogN] [int] NULL ,
[Rpt_Path_Name] [nvarchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
[Proc_Start] [nvarchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
[Proc_End] [nvarchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
[Group_1_Value] [nvarchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
[Status] [nvarchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
[Failure_Reason] [nvarchar] (4000) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
[Warning] [nvarchar] (4000) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
[Export_File_Name] [nvarchar] (4000) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
[Email_To] [nvarchar] (4000) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
[Email_Subject] [nvarchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
[Command_Line] [nvarchar] (4000) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
[User_ID] [nvarchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
[PC_ID] [nvarchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NULL
) ON [PRIMARY]
GO
```

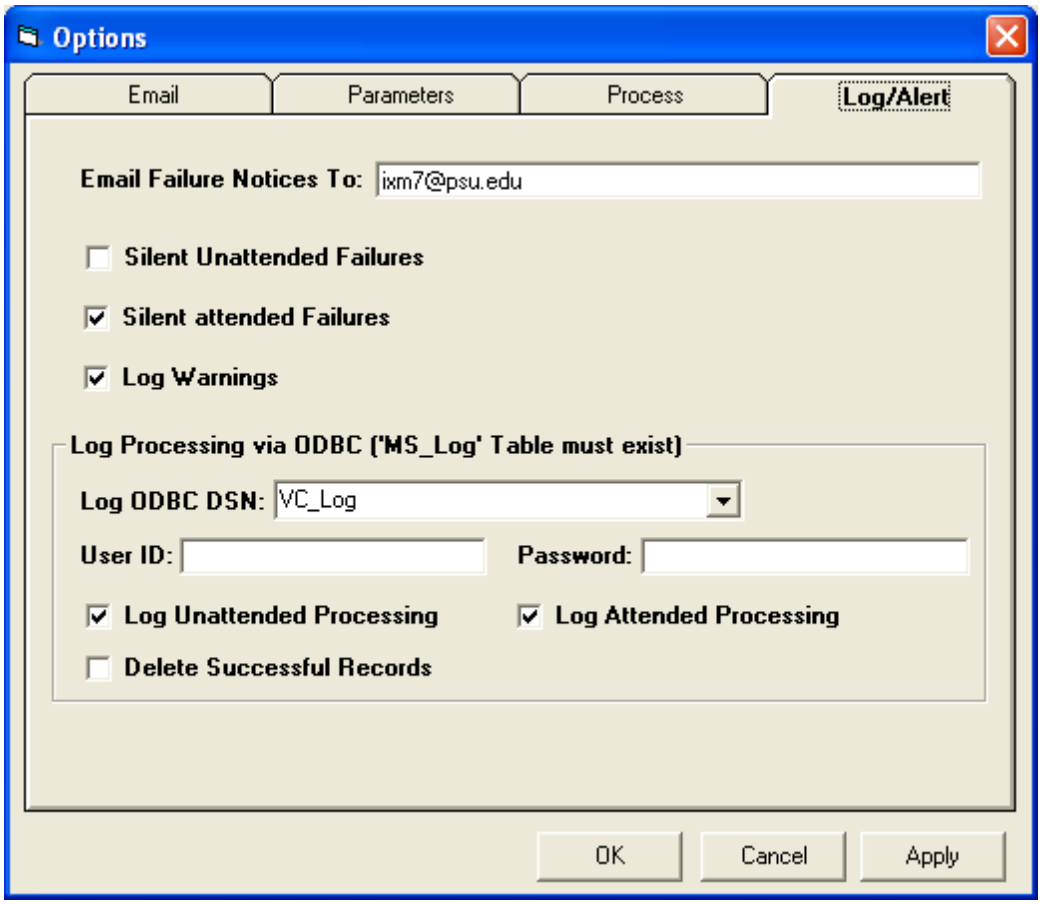
SQL Server notes:

- for recent versions of SQL Server, replace [nvarchar] (**40000**) with [nvarchar] (**MAX**)
- To ensure the User ID you use can access that table, make that user a member of the dbo_owner group in SQL Server.

You can create Crystal reports against this table. You can even schedule these reports in Visual CUT to alert you about bad email addresses, failing, and slow (using the Proc_Start and Proc_End columns) reports.

Process Logging Settings in Options Dialog

The following Visual CUT Options dialog allows you to specify the ODBC Data Source Name (DSN) where the MS_Log table resides and the User ID & Password (stored encrypted), if the data source requires a login.
Checkboxes allow you to control whether logging should occur only for processing triggered via command line, only for processing triggered interactively (user clicks the START button), both, or neither.
You can also specify whether records of successful processing should automatically be deleted (in cases where you wish to record only failures).
Note that Warning messages will not be logged to the database if the "Log Warnings" option is turned off.



Update a Database After Success (After_Success_SQL)

After Visual CUT successfully exported, printed, or emailed a report, you may want to update your database to reflect that information. For example, after bursting invoices to customers, you may want to update the records for these jobs to reflect the date of invoicing or the fact that an invoice was emailed. Besides providing useful information, these columns may also be used to avoid duplicate processing by incorporating them into the record selection formula in your reports.

To update a database after successful processing, you use the **After_Success_SQL** command line argument. The argument structure is as follows:
… **"After_Success_SQL:Type>>ODBC DSN>>User ID>>Password>>SQL Statement"**
or to trigger multiple statements (each using a different ODBC DSN) repeat the 5 elements after a ^^^^ delimiter:
"After_Success_SQL:Type>>DSN1>>User1>>Pass1>>SQL1^^^^Type>>DSN2>>User2>>Pass2>>SQL2"

The parameters (after the "**:**") are separated by a "**>>**" and are as follows:

1. **Type:** the type of success step: **Burst** or **Whole**
10. **ODBC DSN:** The ODBC DSN providing access to the target database. Note that the target database doesn't have to be the same as the one used for the report.
11. **User ID:** Leave blank if no user id is needed to connect to the ODBC DSN
12. **Password:** Leave blank if no password is needed to connect to the ODBC DSN
13. **SQL Statement:** the SQL statement to execute. This typically include embedded references to fields/formulas that Visual CUT would replace with their dynamic values.
    NOTE 1: if the SQL statement is blank, it simply gets skipped (no failure message).
    Note 2: to specify multiple statements simply separate them with a '**;** '

For example, the following command line argument

```
… "After_Success_SQL:Burst>>xTreme>>>>>>Update "JOBS" SET "Processed" = True, "Date Processed" = Date() WHERE "Job Name" = '{@Job}'"
```

Would trigger a SQL statement through the xTreme ODBC DSN, without user id and password, every time a bursting step is completed successfully. The statement would find the matching record in the JOBS table by comparing the Job Name column to the value of a {@Job} formula in the report. If one or more matching records are found, their "Processed" column is set to TRUE, and their "Date Processed" column is set to the current date.

Note that the syntax above conforms to **MS Access**. For other databases you may need to adjust the syntax. For example, MS Access provides a Date() function. MS Access also requires enclosing table and column names with double quotes, and literal strings with single quotes (as done around the value of the {@Job} reference).

SQL Server Example 1

Here is a simple SQL Server example (note the **double quotes around table/column names** and **single quotes around string values**:

```
"After_Success_SQL:Burst>>SQL2>>>>>>UPDATE "TCM93"."dbo"."VC_ship_notify" SET
GF1_DASH_Update = CURRENT_TIMESTAMP WHERE "VC_ship_notify"."gf1_company_code"
= 'PSI' and "VC_Ship_Notify"."GF1_ID_Ord" = '{@Order_ID}'"
```

SQL Server Example 2

Here is an example using SQL Server syntax. In this case, the user needed to INSERT a record into a SQL Server table after each bursting step. The command line argument used was:
"After_Success_SQL:Burst>>CR_DSN>>>>>>{@SQL_Formula}"
No user id or password were specified above because the connection used NT Authentication rather than SQL Server authentication.

The {@SQL_Formula} was placed in Group Footer 1 and it's expression was for it was as follows. Note that you must surround string values with explicit single quotes. This was taken care of by the formulas. For example: `"'" & {@Some_Text} & "'"`

```
'insert into CrTestDatabase.dbo.NewRenewalFees ( ' &
  'DOC_YEAR, ' &
  'BUSINESS_NO, ' &
  'BUS_LIC_NO, ' &
  'CODE, ' &
  'CLASSIFICATION, ' &
  'NUMUNITS, ' &
  'NUMSEATS, ' &
  'RenewalFee, ' &
  'HOCCRDue, ' &
  'RenewalId, ' &
  'HOCCRId, ' &
  'REPORT_ID ' &
') ' &
'SELECT  ' &
  {@Year}              & " as DOC_YEAR, " &
  {@BusNo}             & " as BUSINESS_NO, " &
  {@BusLicNo}          & " as BUS_LIC_NO, " &
  {@Code}              & " as CODE, " &
  {@Classification}    & " as CLASSIFICATION, " &
  {@NUMUNITS}          & " as  NUMUNITS, " &
  "0.00 as NUMSEATS,  " &
  {@RenewFee}          & " as RenewalFee, " &
  {@HOCCRFee}          & " as HOCCRDue, " &
  {@RecIdRenewal}      & " as RenewalId, " &
  {@RecIdHOCCR}        & " as HOCCRId, " &
  {@ReportId}          & " as REPORT_ID"
```

Update a Database Before Report Runs (Before_Report_Run_SQL)

Before Visual CUT runs the report (but just after any Email Capture processes), you may want to update a database. For example, the report may use a temporary table that gets created via your SQL statement. The structure and logic of the **Before_Report_Run_SQL** command line argument is identical to **After_Success_SQL** earlier argument except that:
a) you may not refer to report fields/formulas, and
b) there is no Type (Whole/Burst) element.

The argument structure is as follows:
… **"Before_Report_Run_SQL:ODBC DSN>>User ID>>Password>>SQL Statement"**
or to trigger multiple statements (each could use a different ODBC DSN) repeat the 4 elements after a ^^^^ delimiter:
…"**Before_Report_Run_SQL:DSN1>>User1>>Pass1>>SQL1^^^^DSN2>>User2>>Pass2>>SQL2"**

Call a Web Service after Success (After_Success_HTTP)

After Visual CUT successfully exported, printed, or emailed a report, you may want to trigger a call to a web service. For example, you can use the EzTexting service to send SMS messages without needing to know the carrier associated with the target mobile phone number.

To automate this type of workflow, you can use the **After_Success_HTTP** command line argument. The argument structure is as follows:
… **"After_Success_HTTP:Type>>URL>>Tokens"**

The parameters (after the "**:**") are separated by a ">>" and are as follows:

1. **Type:** the type of success step: **Burst** or **Whole**
2. **URL to call the web service:** this include any arguments expected by the web service
3. Tokens (Optional): if the web service responds with XML text, specifying the names of the XML nodes (separated by '||') allows the process to report back on the returned values of these tokens in the progress dialog (if the process was triggered interactively).

Note: if triggered interactively (command line argument is in the GUI), the progress dialog reports back on the status of these calls. Failures are also logged as warnings if ***Log Warnings*** is turned on (in the Log/Alert tab of the Options dialog).

Sending SMS Messages

For example, using the EzTexting service (500 free SMS messages per month) you can send SMS by following the instructions for constructing a url call. An example of such a url is:
https://app.eztexting.com/api/sending/?user=myUserID&pass=myPass&phonenumber=8141234567&subject=New Ticket&message=Printer 4 is down
a similar example using references to Crystal formulas, and skipping the Subject argument, is:
https://app.eztexting.com/api/sending/?user=UserID&pass=myPass&phonenumber={@Phone}&message={@Message}

The full command line argument (can be placed in the GUI area, in a batch file, etc.).
In a case where the full url is constructed as a {@URL} formula, and the optional Tokens argument is skipped, the command line argument (in a case of **Burst** rather than **Whole**) would simply be:

```
---------------------------------------------------------------
 … "After_Success_HTTP:Burst>>{@url}"
---------------------------------------------------------------
```

This would trigger a separate SMS for each bursting step. Visual CUT automatically inserts a delay (half a second) between each step in order to respect typical limits on message rates.

Trigger Dynamic Batch File after Success (After_Success_Batch)

After Visual CUT successfully exported, printed, or emailed a report, you may want to trigger a batch file to execute follow up processes that depend on a successful completion of the prior process. Or perhaps after a successful bursting of shipment notifications, you wish to send an email to the team responsible for handling invoicing.

To automate these type of workflows, you can use the **After_Success_Batch** command line argument. The argument structure is as follows:
… **"After_Success_Batch:Type>>Batch_File_Path_and_Name"**

The parameters (after the "**:**") are separated by a ">>" and are as follows:
4. **Type:** the type of success step: **Burst** or **Whole**
5. **Batch_File_Path_and_Name:** The path and name of the batch file to trigger

For example, the following command line argument

… **"After_Success_Batch:Whole>>c:\Batch\Ship_Burst_Done.bat"**

Would trigger the **Ship_Burst_Done.bat** after all bursting steps in the current process completed successfully.

Dynamic References to Fields/Formulas within the Batch File

A key feature is that you can embed field/formula names within the batch file just as you can within the Visual CUT 3rd tab options. Visual CUT substitutes the appropriate values for these field/formula names before launching the batch file processing.

Logging to a text file

Since a batch file can write to a text file, **you can use After_Success_Batch to record successful processing to a text file**. For example, the following batch file (C:\Batch_Files\Demo_Log_To_Text_File.cmd) uses @ECHO Off to turn off echoing the command lines to the screen, and then logs some text with references to fields/formulas to a specified text file.

```
@Echo Off
@Echo Sales report for {Product_Type.Product Type Name} in {@Year_Parameter} was processed successfuly {[yyyy]}/{[MM]}/{[dd]} {[hh]}:{[nn]} >> "c:\temp\My Log.txt"
```

This batch file is triggered via a command line argument of:
"After_Success_Batch:**Burst**>>C:\Batch_Files\Demo_Log_To_Text_File.cmd"
and the bursting steps get logged as follows:

```
Sales report for Competition in 2004 was processed successfuly 2015/05/18 07:49
Sales report for Mountain in 2004 was processed successfuly 2015/05/18 07:49
Sales report for Hybrid in 2004 was processed successfuly 2015/05/18 07:49
…
```

:

bursting a report and adding this to the arguments area:

e following using the Visual CUT 11 sample report with a bursting

Table of Command Line Arguments

This table lists the optional command line arguments supported by Visual CUT. The **order of the argument is not important**, but they should **follow the mandatory call to the Visual CUT executable, the execution mode flag (-e or –E) and the path & file name of the rpt file**. The arguments should be **enclosed in double quotes** and be **separated by a space**. For example:
"C:\Program Files\Visual CUT\Visual CUT.exe" **-e** "C:\Program Files\Visual CUT\Visual_CUT.rpt" "Skip_Recent:2880"

Argument	Purpose	Example
Data Source Options		
Attempt_Logon_Without_Password	Override the ini setting	"**Attempt_Logon_Without_Password**:True"
User_ID	User ID	"**User_ID**:dba"
Password	Password	"**Password**:sql"
ODBC_DSN	Override all ODBC Data Sources	"**ODBC_DSN**:Test_Server"
ODBC_DSN_From_To	Override one ODBC Data Source	"**ODBC_DSN_From_To**:Production_Server>>Test_Server"
Table_From_To	Override Table Location (.csv)	"**Table_From_To**:Risk.csv >>Risk2.csv||Old.csv>>New.csv"
Database_Path	Set path to Access/Excel/Pervasive	"**Database_Path**:C:\Prop1\FILE.DDF>>c:\Prop2\File.DDF"
Oracle_Server	Oracle Server (Native Connection)	"**Oracle_Server**:Test1"
Connect_To_SQLOLEDB	SQL Server OLE DB Data Source	"**Connect_To_SQLOLEDB**:Server2>>Northwind>>False"
Report Parameters/Formulas		
Parm1, Parm2, Parm3…	Set/Override saved Parameter Values	"**Parm3**:9/15/2005"
Set_Formulas1	Set Formula Expressions	"**Set_Formulas1**:{@^Name}>>>[dblq]Jane[dblq]|||{@^TwoAndTwo}>>>2+2"
Printing Options		
Printer	Printer Destination	"**Printer**:\\server10\hp04"
Printer_Only	Printer Destination	"**Printer_Only**:\\server10\hp04"
Printer_Burst	Print Bursting Destination	"**Printer_Burst**:\\server10\hp04"
Printer_Burst_Only	Print Bursting Destination	"**Printer_Burst_Only**:{@Group_Printer_Name}"
Print_Copies	Number of Copies to Print	"**Print_Copies**:{@Label_Quantity}"
PDF_PRINT	Print a merged PDF File	"**PDF_PRINT**:c:\temp\Result.pdf>\\server10\hp04"
PDF_PRINT_SPLIT	Print PDF file across paper trays	"**PDF_PRINT_SPLIT**:Test.pdf>1::\\s1\HP::Top||2to99::\\s1\HP::Bottom"
PDF_PRINT_SPLIT_TAG	Print PDF file across paper trays	"**PDF_PRINT_SPLIT_TAG**:c:\temp\Test.pdf>\\s1\HP"
Export Options		
Export_Format	Export Format	"**Export_Format**:Excel 97"
Export_File	Export File	"**Export_File**:c:\temp\Invoice_for_{CustName}.xls"
Export_Mode	Export Mode (**Burst** or **Whole** or blank)	"**Export_Mode**:Burst"
Release_Shared_File	Avoid export failure when file is in use	"**Release_Shared_File**:True"
After_Export_Delay	Delay in Milliseconds	"**After_Export_Delay**:250"

Table of Command Line Arguments (Contd.)

Argument	Purpose	Example
Email Options		
Email_To	Email To	"**Email_To**:{@Cust_Email}" or "**Email_To**:File:c:\Reps_Emails.txt"
Email_From	Email From	"**Email_From**:ixm7@psu.edu"
Email_Reply_To	Email Reply To	"**Email_Reply_To**:ixm7@psu.edu"
Email_CC	Email Copy To	"**Email_CC**:ixm7@psu.edu"
Email_BCC	Email Blind Copy To	"**Email_BCC**:ixm7@psu.edu"
Email_Attach	Email Attachment(s)	"**Email_Attach**:c:\Invoice_for_{CustName}.xls;c:\Instructions.pdf"
Email_Subject	Email Subject	"**Email_Subject**:Invoice for {Product_Name}"
Email_Message	Email Message (can be HTML)	"**Email_Message**:<html>Hi {customer_first_name}, ...</html>"
Email_Message_Save	Save body to a Text/HTML file	"**Email_Message_Save**:W:\Dashboards\Sales_Dashboard.html"
Email_Mode	Email Mode (**Burst** or **Whole** or blank)	"**Email_Mode**:Burst"
Email_Priority	Priority (Highest/High/Normal/Low/Lowest)	"**Email_Priority**:Highest" or "**Email_Priority**:{@Priority}"
Email_Header	Custom Email Header(s)	"**Email_Header**:X-Sensitivity: 3" or "**Email_Header**:{@email_header}"
Email_SMTP_Server	SMTP Server address	"**Email_SMTP_Server**:127.0.0.1"
Email_SMTP_Port	SMTP Communications Port	"**Email_SMTP_Port**:26"
Email_User_ID	Authenticate to SMTP Server	"**Email_User_ID**:ixm7"
Email_Password	Authenticate to SMTP Server	"**Email_Password**:my_secret_password"
Email_Delay_MilliSeconds	Delay Email Processing	"**Email_Delay_MilliSeconds**:500"
Email_Send_Encrypted	Should emails be Encrypted	"**Email_Send_Encrypted**:True "
Email_Send_Signed	Should emails be Signed	"**Email_Send_Signed**:True "
Email_StartTLS	SMTP Server Requires StartTLS	"**Email_startTLS**:True"
Email_Bounce_Address	Set destination for bounced emails	"**Email_Bounce_Address**:ido@MS.com"
Email_SMTP_Domain	For SMTP integrated authentication	"**Email_SMTP_Domain**:Your_Domain"
Email_Outgoing_Folder	For background (queue) emailing	"**Email_Outgoing_Folder**:C:\VC\Outgoing"

Table of Command Line Arguments (Contd.)

Argument	Purpose	Example
PDF Processing (in the order they are processed if specified in the same command line)		
PDF_Export_Options	Use MS Word as Export Engine	"PDF_Export_Options:Tagged and PDF/A"
PDF_FORM	Fill Form Fields in a PDF File	"**PDF_FORM**:c:\temp\MyForm.pdf>False"
PDF_Bookmarks_Open_Levels	Which Bookmarks Levels to Expand	"**PDF_Bookmarks_Open_Levels**:**0**"
PDF_LinkToWeb	Add web or email hotspot	"**PDF_LinkToWeb**:c:\test.pdf>1>99>40>60>50>10>http://www.IBM.com>>>0"
PDF_AddImage	Add an image with optional hotspot	"**PDF_AddImage**:{@pdf_file}>1>99>40>60>50>10>{@Link}>{@Logo}>1"
PDF_From_TIFF	Import multi-page TIFF into PDF	"**PDF_From_TIFF**:c:\tmp\Invoice_{@Inv_N}.pdf>{@Tiff_File}>37>True>True"
PDF_MERGE	Merge PDF Files	"**PDF_MERGE**:c:\temp\File1.pdf,c:\temp\File2.pdf>c:\temp\Result.pdf"
PDF_MERGE_Files_to_Layers	Turn pdf files into layers in a single file	"**PDF_MERGE_Files_to_Layers**:PDF_File_List>PDF_File_Target"
PDF_Insert_BackPage	Add standard content after each page	"**PDF_Insert_BackPage**:c:\temp\{Invoince_N}.pdf||c:\Legal_Page.pdf"
PDF_Bookmark_Tags	Add Bookmarks based on formulas	"**PDF_Bookmark_Tags**:c:\temp\{@customer}.pdf"
PDF_TOC	Add Table of Contents to PDF File	"**PDF_TOC**:1>40>11>6>5>2>c:\temp\Sales.pdf"
PDF_PAGE_N	Add Page Numbers to PDF File	"**PDF_PAGE_N**:2>10>10>11>Bottom>Center>Page_NofM>Sales.pdf>Helvetica"
PDF_PRINT	Print (stapled) a merged PDF File	"**PDF_PRINT**:c:\temp\Result.pdf>\\server10\hp04"
PDF_Print_Mode	Set Quality/Speed of Print	"**PDF_Print_Mode**:2"
PDF_PRINT_SPLIT	Print PDF file across paper trays	"**PDF_PRINT_SPLIT**:Test.pdf>1::\\s1\HP::Top||6to99::\\s1\HP::Bottom"
PDF_PRINT_SPLIT_TAG	Print PDF file across paper trays	"**PDF_PRINT_SPLIT_TAG**:c:\temp\Test.pdf>\\s1\HP"
PDF_FORM_Tags	Create Form Fields in a pdf file	"**PDF_Form_Tags**:c:\temp\Purchasing_List.pdf"
PDF_Flatten	Flatten Annotations and Form Fields	"**PDF_Flatten**:c:\temp\Forms*.pdf>>myPassword>>c:\temp\Results\"
PDF_Embed	Embeds file(s) in a PDF	**"PDF_Embed:c:\Main.pdf<<c:\S1.pdf:: application/pdf::Spec1||c:\D1.jpg::image/jpg::Draw1"**
PDF_Link_Tags	Add Link/Image based on formulas	"**PDF_Link_Tags**:c:\temp\{@customer}.pdf"
PDF_Link_Tags2	Embeds and Links to Internal Files	"**PDF_Link_Tags2**:c:\temp\{@customer}.pdf"
PDF_Auto_File_Link	Detect file references and add links	"**PDF_Auto_File_Link**:c:\In.pdf>>.>>.pdf;.mp3>> (>>0>>)>>0;255;0>>True>>"
PDF_Auto_File_Link_Tokens	Detect file references and add links	"**PDF_Auto_File_Link_Tokens**:..\media\MM##.wmv||..\Docs\DOC##.pdf"
PDF_Save_As	Saves a PDF File to Image File(s)	"**PDF_Save_As**:Test.pdf>Test.bmp>BMP>72"
PDF_Properties	Set PDF Document Properties	"**PDF_Properties**:pdf file(s)>Author>Title>Subject>Keywords>Creator>Producer >PageMode>StartPage>Zoom"
PDF_Build_Index	Builds Index for all pdf files in folders	"**PDF_Build_Index**:c:\temp\dir1||c:\temp\dir2>c:\temp\Index.pdx>"
PDF_Add_Index	Adds an Index File Reference	"**PDF_Add_Index**:pdf file>index file>Index Label "
PDF_Add_Media	Add embedded Multimedia	"**PDF_Add_Media:**infile>>outfile>>media_file>>1>>X>>Y>>W>>H >>Sound>>"
PDF_Compress	Reduce Size of PDF File	"**PDF_Compress**:PDF_File>>PDF_File_Target"
PDF_Split_By_Bookmarks	Split a PDF based on bookmarks	"**PDF_Split_By_Bookmarks**:PDF_File>>Level>>Split File"
PDF_Split_Tags	Split a PDF based on formulas	"**PDF_Split_Tags**:PDF_File>>False>>False>>True"
PDF_Sign	Add Digital Signature	"**PDF_Sign**:Input pdf>>Output pdf>>Open Password>>Sig Field>>pfx File>> Password>>Reason>>Location>>Contact
PDF_PROTECT	Encrypt/Protect a PDF File	"**PDF_PROTECT**:Owner_Pass>User_Pass>1>1>0>1>1>c:\Sales.pdf"

PDF_Linearize	Web-enable for faster load in browser	"**PDF_Linearize**:PDF_File>>PDF_File_Target>>Password"

Table of Command Line Arguments (Contd.)

Argument	Purpose	Example
EXCEL Processing		
XLS_Print_Setup	Set Excel Print Options	"**XLS_Print_Setup**:c:\temp\{Customer.Country}.xls>>>1>1>>>>>>>>>"
XLS_Save_As	Save Excel File as a PDF file	"**XLS_Save_As**:c:\temp\Invoice.xlsx>c:\temp\Invoice.pdf>PDF>0>0>0"
XLS_AutoFilter	Auto Filter and/or Freeze Panes	"**XLS_AutoFilter**:True" or "**XLS_AutoFilter**:A1>>A2"
XLS_AutoFit	Fit Column Widths in Excel	"**XLS_AutoFit**:True" or "**XLS_AutoFit**:MaxColumnWidth>>31>>False"
XLS_to_XLSX	Convert multi-tab **xls** to 1-tab **xlsx** file	"**XLS_to_XLSX**:c:\temp\In.xls>>c:\temp\Out.xlsx>>True"
XLS_Pivot_Table	Generate an Excel Pivot Table	**"XLS_Pivot_Tablet**:…see user manual for arguments…**"**
XLS_Range_Insert	Insert formula content in named ranges	"**XLS_Range_Insert**:c:\template.xls>>c:\target.xls"
XLS_Replace	Replace. Activate formula expressions.	"**XLS_Replace**:c:\temp\Input.xls||Output.xlsx||^=>>=::ab>>cd||"
XLS_Refresh	Refresh Queries and Pivot Tables	"**XLS_Refresh**:c:\source.xls>>c:\target.xls>>Sleep Seconds"
XLS_Run_Macro	Run Excel Macro	"**XLS_Run_Macro**:c:\source.xls>>c:\target.xls>>Macro_Workbook>>Macro_Name"
XLS_Transfer_Tabs	Transfer and Rename Tabs	"**XLS_Transfer_Tabs**:[{@F1}.xls||Tab1::Tab2]>>[{@F2}.xls||{@Tab1}::{@Tab2}]"
XLS_Split_Tabs	Save each Tab to PDF or Excel file	"**XLS_Split_Tabs**:{@F1}.xlsx||PDF||{@F1}_[Tab_Name].pdf||Landscape||True"
XLS_Protect_Worksheets	Protect content against viewing/editing	"**XLS_Protect_Worksheets**:Source_File>>Target_File>>Password"
XLS_Protect	Password Protect Excel Files	"**XLS_Protect**:File_List>Password"
MS Word Functionality		
WORD_Replace_Tags	Replace Formula tags with Values	"**WORD_Replace_Tags**:c:\temp\Template.doc>c:\temp\Contract_{Cust_Name}.doc"
WORD_Replace_Format	Replace Formatting	"**WORD_Replace_Format**:c:\A.docx>c:\B.docx>Strikethrough>DoubleUnderline"
WORD_Print	Print a Specified MS Word File	"**WORD_Print**:{@Word_Doc}>>Default>>{@Copies}"
WORD_Print_Watermark	Print with dynamic Watermarks	"**WORD_Print_Watermark**:c:\Invoice.doc>>Default>>>>{@Copies}>> Copy [[N]] of [[M]]>>Calibri>>False>>False>>12>>1.3>>7.5>>315>>0.5>>"
WORD_Protect	Restrict Viewing / Modification	"**WORD_Protect**:{@inFile}>>{@inFile}>pwd1>>pwd2>Allow_Only_Form_Fields"
WORD_Save_As	Save a Word document to PDF	"**WORD_Save_As**: {@file_name}.doc>{@file_name}.pdf>0>0>0>1>0"
TEXT/HTML Processing		
TXT_Split_Tags	Splits a text file into multiple files	"**TXT_Split_Tags**:c:\temp\File.txt>>Replace"
TXT_Merge	Merges Text Files	"**TXT_Merge**:c:\temp\File1.txt,c:\temp\File2.csv>c:\temp\Result.csv>0"
TXT_Replace	Replace Content in a Text File	"**TXT_Replace**:c:\temp\Input.csv||Output.csv||Chr(10)>>::Chr(13)>>||End"
TXT_Remove_Short_Lines	Remove blank or short lines	"**TXT_Remove_Short_Lines**:InFile||OutFile||Max_Length_To_Remove"
TXT_Replace_Tokens	Replace Content in a Text/HTML File	"**TXT_Replace_Tokens**:File||OutFile||Start^^End^^replace^^Location^^Options ::Start2^^End2^^replace2^^Location2^^Options2||Global_Options"
TXT_DeGUID_png	Clean png file refs in HTML Exports	"**TXT_DeGuid_png**:HTML_File||Options"

Table of Command Line Arguments (Contd.)

Argument	Purpose	Example
Miscellaneous		
After_Burst_Batch	Interweave Reports Output	"**After_Burst_Batch**:C:\VC_Prod_Type.bat"
After_Export_Batch	Trigger Batch file after export	"**After_Export_Batch**:C:\temp\Archive_to_RAR.bat"
Before_Report_Run_SQL	Update DB Before Report Run	"**Before_Report_Run_SQL:**MyDSN>>>>>>Update "JOBS" SET "Processed" = FALSE WHERE "Emailed_To" = 'Not Yet' "
After_Success_SQL	Update DB after Processing	"**After_Success_SQL:**Burst>>MyDSN>>>>>>Update "JOBS" SET "Processed" = True, "Date Processed" = Date() WHERE "Job Name" = '{@Job}'"
After_Success_HTTP	Trigger web service	"**After_Success_HTTP:**Burst>>{@URL}"
After_Success_Batch	Trigger Batch File	"**After_Success_Batch:Whole**>>c:\temp\Ship_Notifications_Burst_Done.bat"
Before_Export_Batch	Trigger Batch before export	"**Before_Export_Batch**:C:\temp\Archive_Previous_Exports.bat"
Encrypted_Password_Set_Entry	Set Named Password	"**Encrypted_Password_Set_Entry**: H:\DataLink_Viewer.ini>>Options>>Encrypted_Password_FTP>>sesame"
SQL_Extract_Files	Extract Files from Database	"**SQL_Extract_Files**:SG>>>>>>Picture>>c:\temp\>>Id>>.png>>Select * From Students>>False"
Skip_Recent	Avoid duplicate processing	"**Skip_Recent**:2880"
JOB_STATUS_ID	Control Job Status File Name	"**JOB_STATUS_ID**:4523"
FTP_Download	Download files from FTP server	"FTP_Download:Passive_1>>www.IBM.com>>milletso>>Encrypted_Password_SFTP>>/public/test/*.pdf>>c:\temp\"
FTP_Upload	Upload Files to FTP Server	"**FTP_Upload**:Passive_1>>ftp.abc.com>>user>>pass>>{@Export_File}>>/public_html/Download"
SFTP_Upload	Upload Files to **S**FTP Server	"**SFTP_Upload**:22>>myserver.com>>PW>>user>>pass>>>>>>{@ExportFile} >>" "
Use_Saved_Data	Use Saved Data in report	"**Use_Saved_Data**:True"
Use_Saved_Data_Recent	Use Saved Data if Not Older than N Minutes	"**Use_Saved_Data_Recent**:60"
Write_INI_Location	Writes ini File Location to specified file	"**Write_INI_Location**:c:\temp\VC_ini_Location.txt"
Main_Files_Folder	Specify Location of key mdb & ini files	"**Main_Files_Folder**:path to folder holding DataLink_Viewer.ini & Visual CUT.mdb "
ZIP_Files	ZIP & Password Protect Files	"**ZIP_Files**:File_List>Password>ZIP_File"

Update History

Version 6.9002: Entered Testing December 13, 2015

- When using the "Print_Copies:" argument for direct printing (not via PDF_Print), you can now set custom text on each print copy (such as 'Copy 1 of 2', 'Copy 2 of 2'). Note that the print quantity can be dynamically controlled by a Crystal field/formula. See 'Setting Custom Text for Each Print Copy' and this demo image.

- During interactive exporting, a "Printer:" directive placed in the Arguments area of the GUI now also triggers printing.

- Increased the width of the Copies option to reflect cases where a field/formula is dragged into that option.

- The HTML editor now avoids adding unnecessary style directives to the email message body ("style=""BORDER-TOP-COLOR: ; BORDER-LEFT-COLOR: ; BORDER-BOTTOM-COLOR: ; BORDER-RIGHT-COLOR: ").

- When command line processing of a report includes an **Email_Get** directive, if the number of newly captured emails that got inserted into the database is zero, the process now aborts without wasting time on processing the report (since there are no new records to process). You may override that new default behavior by setting an ini file option. See ' Triggering Email Capture'.

- The Email To, CC, and BCC labels now act as buttons that launch a dialog for viewing and selecting previously used emails (individual as well as grouped emails). The list sorted by frequency of past use. You can use a checkbox or a double-click to Select/Unselect addresses. Addresses already populated in the clicked field display as Selected. The dialog supports searching. See demo image.

Fixes

- Fixed a report refresh/reload scenario causing the first group to lose dynamic field/formula values.

- Fixed an issue with cloning rpt files when the internal settings database is redirected to certain types of DBMSs.

Version 6.9001: Released November 21, 2015

Key Features

- **Visual CUT 11 can now delegate exporting/bursting to DataLink Viewer 2011 in order to support Crystal 2008/2011/2013 features**: a) mixed portrait & landscape sections, b) Dissociate Formatting Page Size and Printer Paper Size, c) Auto-Arrange option for chart layouts, d) 'New Page After N Visible Groups', e) Calculated CrossTab members, and f) embedded Flash objects. See 'Delegating Exporting/Bursting to DataLink Viewer 2011' and this demo image.

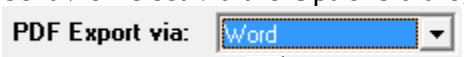
- Added **XLS_Range_Insert_File** command line argument to insert Excel (Data Only) exports into pre-formatted excel templates. This **allows inserting larger data sets** (compared to XLS_Range_Insert, which uses values in Crystal formulas). It also supports **appending data and cloning template formulas, conditional formats, and sparklines**. See 'Inserting File Exports into Excel Templates.' See demo image.

- Added **After_Success_HTTP** command line argument. This **allows triggering calls to web services such as sending SMS to mobile phones**. See '*Call a Web Service after Success (After_Success_HTTP)*' and '*Sending SMS Messages*'.

- **XLS_Save_As can now convert pivot table to HTML**. This allows **embedding Pivot Tables (generated via XLS_Pivot_Table) inside auto-refreshing web dashboards and inside email message bodies**. See '**Embedding Pivot Tables in Email Message Body**' and this demo image.

- **Parameter fields are now listed in the fields/formulas area for dragging into processing options**. For example, if the first parameter in the main report is called {?Year} it would be listed as **{?01?Year}**. The 01 indicates the position of the parameter in the list of parameters. **This avoids the need to create formulas for referencing parameter values in processing options. It also makes it easy to know what parameter is referenced by command line arguments such as "Parm1:..."** See demo image.

- Added **PDF_Export_Options** command line argument and a global option to use MS Word as the pdf export engine to overcome **font problems** and support **PDF/A** output. The global behavior is set via the Options dialog:

  An ellipsis button to the right of the PDF export format choice launches a **PDF Export Options** window (see image) allowing you to see the global setting, and generate a command line argument to override it for a specific report. See 'Export to PDF via MS WORD.'

- Visual CUT now **checks email addresses for valid structure before interactive start of processing**. The validation process takes into account the dynamic values that would be populated into email addresses during actual processing and bursting (field/formula references, file references to text distribution lists, ODBC queries for email distribution lists). Detected problems are presented to the user with an option to continue or abort processing.

This is particularly useful when bursting emails to many recipients because it allows users to catch problems before the process runs. The Options dialog (Email 2 tab) provides a checkbox allowing users to turn off this option.

- Added **Before_Report_Run_SQL** command line argument to update a database before Visual CUT runs the report (but just after any Email Capture processes).

- **Added options to set date parameters to the first Day of Week before/after a date constant**. For example, the first Monday of the previous month. See *'Adjusting Data Constants for Day of Week*.'

- Added Word_Print_WaterMark command line argument, allowing printing of exported Word files with dynamic watermarks for each printed copy (e.g. "Copy 1 of 4"). See sample image.

- If the report uses an ODBC DSN (including cases where that DSN was specified via a command line argument or via the DSN choice user interface), an **{[ODBC_DSN:]}** object is added to the fields/formulas area for dragging into processing options. This allows emails and other processing options to reflect the actual data source name used by the report. See image.

- The Search & Replace dialog for report paths is now expanded to allow targeting of five other categories of saved settings. See '*Changing Login, Report Paths & Other Settings without Previewing*' and this image. This is useful when you need to **globally change saved settings for emails, export files, attachments, arguments, or parameters**. You may block some or all of these categories (see '*Disabling Find & Replace Categories*').

- **Temp files `(~*.tmp, *}.rpt)` older than 7 days are now deleted upon exiting Visual CUT**.

- Added **TXT_Split_Tags** command line argument allowing fast splitting of a single text file into multiple files based on tags embedded within the text file.

Excel Features

- Added **XLS_Refresh** command line argument to refresh external data used by Excel queries and pivot tables.

- Added **XLS_Split_Tabs** command line argument to generate a separate PDF or Excel file from each excel tab.

- When using **XLS_AutoFilter** to freeze panes in a location deep into an excel worksheet, top rows and left columns no longer become hidden.

- You can now specify the excel file for **XLS_AutoFilter** processing (in the past, only the exported file could be targeted for auto-filter and/or freeze panes options).

- **XLS_Save_As** can now **save to CSV format**. This is **useful for reports that contain subreport** because such scenarios don't export well directly to CSV.

- **XLS_Pivot_Table** operations now handle cases where wrong number of arguments were provided as a failure rather than as a warning. This ensures the user is aware of the problem.

- Visual CUT can now handle **multiple XLS_Pivot_Table arguments**. This means that **a single excel data export can automatically generate multiple pivot tables**.

PDF Features

- Tag formulas for **PDF_Bookmark_Tags now support an optional argument to control the vertical margin above the location of the tag used as the bookmark page location target**. This allows you to override the default of 20 millimeters vertical margin.

- Added **PDF_Flatten** command line argument. This allows flattening all form fields and annotations in a list (wild cards are supported) of pdf files.

- **PDF_Link_Tags** can now handle any type of link such as '**https://**' and '**Tel:**' links.

Email Features

- **When an email failure occurs, a 1-sentence diagnostic is now added to a) the progress window, b) email log, and c) email failure alert.** For example, "Unable to establish a TCP or TLS connection to the SMTP server." This facilitates troubleshooting by avoiding the need to review the verbose email log.

- **Email queuing is now faster** and avoids creating/renaming/moving a temp .eml file.

- **Preview mode in email message HTML editor now displays dynamically referenced images**.

- Added a user manual section on "**Embedding Report as Image in Email Message Body**". See image.

- Fixed image file embedding inside an email message when the path contains spaces.

- **Added group number to the end of .eml file names**.

- The **Email_Failure_Notices_From** option can now be set through the Log/Alert tab of the Options dialog. This is useful when the target address is not allowed as a sending address.

- When VC_Skip_Email is used, the test for missing attachments is now also skipped.

- Added a **Disable_Email_Statusbar_GUI** option (for Master_DataLink_Viewer.ini) to hide the email queue panels in the status bar.

- Added a **Disable_Email_Log_Activity_GUI** option (for Master_DataLink_Viewer.ini) to disable the 'Log Email Activity' checkbox and hide the Notepad button to open the log file.

Other Features

- Auto-conversion from mapped to UNC path now also applies when the user select multiple reports loading into the report grid and when copying .rpt file/settings.

- When saving command lines to a batch file via the scheduling string dialog, Visual CUT now auto-detects cases where a conversion is needed to the local **character set**. This is needed because, for example, a European PC using code page 480 encodes **ü** differently than an American PC using code page 437.
  Also, for cases where you need to force the use of a specific code page (consultant working for a foreign customer), added a **Batch_File_Save_CharSet** option to DataLink_Viewer.ini.
  Note: this makes a difference only when a command line contain special characters.

- When the Visual CUT database is redirected via a connection string, the Version Info window now reflects that info. For example: ***VC Database via Connection String [Server=Serv1\sql2]***.

- A new **Encrypted_Password_Set_Entry** command line argument allows administrators to **automate the saving of Encrypted Passwords to targeted ini files**. For detail, see '*Setting Encrypted Password Entries*.'

Fixes

- Fixed handling of application load failure scenario that could cause a hanging process.

- Fixed FTP_Upload issue caused by trying to create target folders when they already exist.

- Fixed interactive use scenario when a parameter is removed from a previously saved report.

- Added skip logic for folder testing when Printer (Specified) export uses a printer name that starts with '\' character.

- Fixed a bug in **After_Success_Batch** processing with *Burst* option. Also, added a description to the user manual of how After_Success_Batch can be used to log processing to a text file.

- Fixed a problem caused by exporting and then navigating to Preview tab and back again.

- Fixed [VC_NULL] handling for subreport parameters.

- Fixed duplicate processing of some command line arguments in cases where multiple export file names are specified (for example, "c:\temp\Sales.xlsx;c:\temp\Sales.pdf").

Version 6.8001: Released November 16, 2014

Web Dashboard and FTP/SFTP Features

- When exporting to HTML 40, the options button launches a **Web Dashboard Expert** window allowing you to set various options such as a) adding an auto-refresh behavior, b) Setting hyperlinks to launch to new tabs, c) Specifying tab titles and icons, d) Removing GUIDs from referenced.png image files, and e) SFTP Uploading HTML and Other files to the web server. For detail, see *Web Dashboard Expert*
  A web dashboard using that approach is available at:
  http://academicweb.psu-erie.bd.psu.edu/laboccupancy/Labuse.html

- FTP_Upload connection failure messages now include more detail.

- FTP_Upload and SFTP_Upload can now handle file names containing commas.

- FTP_Upload and SFTP_Upload now create missing target folder levels if the target directory doesn't exist on the server.

- Added **FTP_Download** command line argument.

- Relaxed timeout limits for SFTP_Upload in order to handle slow or very busy servers.

PDF Features

- Added a PDF_Insert_Pages_Tags command line argument allowing Crystal formulas acting as invisible tags to specify image files for insertion as pages following the location of the tag. This is useful in cases where related scans or other images need to be inserted in the middle of PDF exports. For detail, see: *Importing Image Files as New Pages (Tag Approach)*.

- **PDF_Form** argument can now use Crystal formula values to fill not only **AcroForm fields** but also **Adobe LiveCycle forms fields**.

- **PDF_Save_As** can now save also to **EMF+**, **HTML5**, and **G4 TIFF** formats.

- The ini entries used to control the layout of the Table of Contents generated using **PDF_TOC** can now refer to report field/formulas. This allows, for example, a different image to be used for the header depending on data in the report.

- When embedding files inside PDF files using **PDF_Link_Tags2**, you can now ignore (skip) missing files by using a lower case "embed" argument in the tag formula.

- Added an optional **Owner_Password** optional argument for **PDF_Merge**. This is useful in cases where one or more of the source files is password protected. All the encryption settings (owner and user passwords, and protection settings) of the last protected file in the list of source files would be applied to the resulting merged document. This allows Visual CUT to **add new information to existing but protected pdf files.**

- Updated **PDF_Linearize** logic.

- Added **PDF_Print_Mode** command line argument to **set printing quality/speed**.

Excel Features

- Special Excel Tab processing (**Tab!**, **TabInOldFile!**, **TabInOldFile_Replace!**, and **TabInNewFile!**) can now be used for any of the export files when specifying **multiple export files** (separated by semi-colon).

- Fixed **XLS_Pivot_Table** compatibility issue with Excel 2013.

- **XLS_Pivot_Table** now accepts an optional Range Name argument.

- **XLS_Pivot_Table** now supports an additional Options argument. This optional argument may support multiple directives in the future but is currently limited to Report_Layout=**Compact**, Report_Layout=**Outline**, or Report_Layout=**Tabular**. **Tabular** uses field names instead of generic 'Row Labels' and 'Column Labels.'

- **XLS_Pivot_Table** can now **target an existing worksheet and cell** for the PivotTable insertion.

- **XLS_Run_Macro** now avoids saving the source workbook after the process runs if the Target Workbook argument is left blank.

- **XLS_Range_Insert** can now insert Crystal formula values into hidden Excel tabs.

- **Added XLS_Replace command line argument**. The user manual explains how **this can be used to export formula expressions and activate them in the exported workbook**.

Email Features

- Added an option (Email 2 tab in Options dialog) to check email addresses and trigger a detailed failure message when malformed addresses are detected.

- Fixed a problem caused by thousands of eml files found in the Outgoing or Undeliverable folders and a scheduled process that completes very quickly. This might cause a "hanging" Visual CUT process.

- Email alerts about processing failures now include the User_ID value if specified via a command line argument. This helps when the rpt is used for multiple databases.

- Fixed blank and [VC_Blank] value handling for email-related command line arguments.

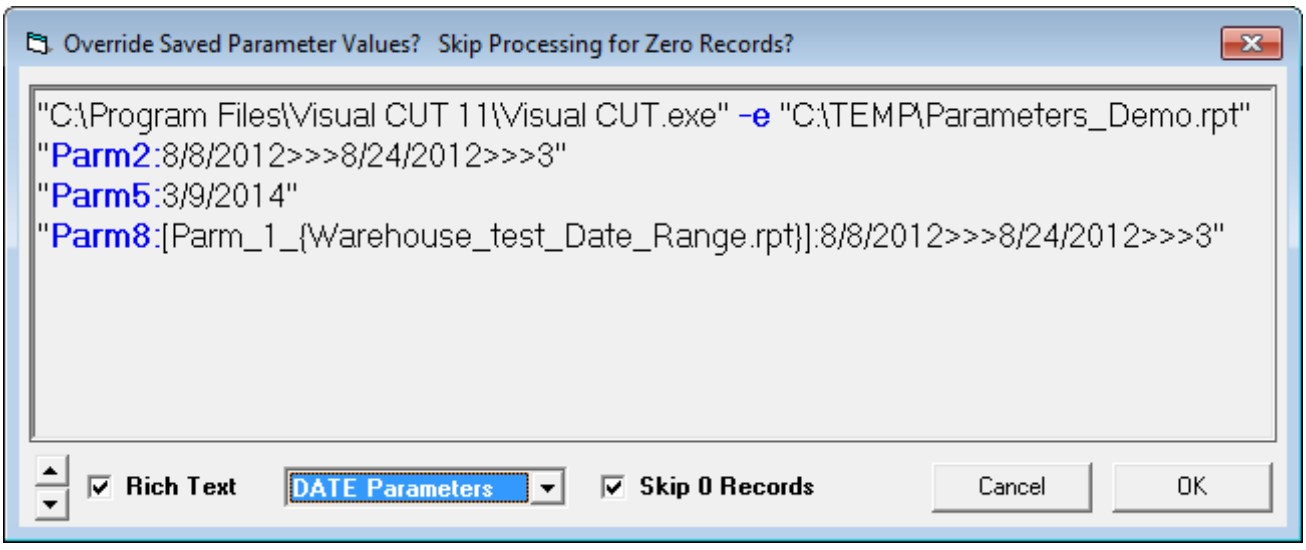
- Added a user manual section on "smtpQ Service Failure Action Properties."

- HTML Editor for email messages now supports UNC paths for inserted image files.

- Added a Save_Attachments_To_Unique_File_Names option to Email Capture directives (ini sections). When capturing incoming emails and downloading attachments to a specified folder, this avoids overwriting prior attachments with the same name. Instead, 4 characters are added to the new attachments to avoid name conflicts. For detail, see "*Email Get Directive Sections*."

Other Features

- The ... button to the right of the scheduling string now starts a **dialog with options to control zero records action (skip or not skip processing), and to override saved parameter values in the initial command line**. If the report has Date or DateTime parameters, the dialog defaults to including them. You can then edit the static values in Notepad, and replace them with Date Constants such as Yesterday, Start_Month_Minus_1, etc. (see Date Constants section in the user manual).



- You can now use "**ParmN:[?]**" command line arguments to indicate that VC should **prompt the user for certain parameter values**. This is useful when VC is called from a command line and the user needs to interactively override saved parameter values. For detail, see "Request User Input for Certain Parameters."

- The Process tab in the Options dialog now has a button called '**Encrypt & Save Password**'. It allows you to **centralize & protect passwords by avoiding specifying them directly inside command line argument**. Instead, you can name, encrypt and store the passwords inside DataLink_Viewer.ini. For detail, see "Referring to Saved Encrypted Passwords."

- Added **SQL_Extract_Files** command line argument. Databases can store files (images, spreadsheets, pdf, audio, etc.) within a database as binary column types. Certain workflow scenario may require extracting these files to the file system for further processing (merging, emailing…). This allows Visual CUT to automate such processes.

- Added **Use_Saved_Data_Recent** command line argument. For example, "**Use_Saved_Data_Recent:60**" tells Visual CUT to use saved data if it is not older than 60 minutes. Otherwise, fresh data is retrieved.
  A typical use scenario is to export a report over itself (refreshing the saved data inside the report) as well as exporting and/or emailing a requested format (Visual CUT can export to multiple semi-colon separated files). Future requests for the same output avoid repeated data retrievals if the saved data is fresh enough.
  Note: if the source rpt/rpz file is included in the semi-colon separated list of export files, and the process uses saved data (due to this command line argument), then the export of the source rpt/rpt file over itself is skipped if it's not the first file in the list.

- **Added the ability to change folder location for ACT! (*.pad) data source files**. See "Changing Folder Location for Access/Excel/Pervasive/ACT! Files."

- Added {%UserName%} to the fields/formulas area for dragging into processing options.

- The **Main_Files_Folder** option (ini setting as well as the command line argument) now supports dynamic substitution for any combination of the following **environment variables**: %UserName%, %UserProfile%, and %Appdata%

- **Added TXT_Replace_Base64 command line argument.** This allows automating the process of inserting Base64-Encoded Files Inside Text exports. For example, you can export invoice data to an XML file containing a reference to an image or pdf file. VC then replaces the reference to the file with the content of that file encoded as Base64 (allowing conversion from Binary to Text representation). The XML file with the embedded image file can then be transmitted to a business partner using SFTP_Upload. For detail, see: *Inserting Base64-Encoded Files Inside Text/XML*.

- **TXT_Merge** can now handle cases where the files are merged into the first file (first file is the target file). Also, there is no longer a need to specify the path to the target file if the path is the same as the first source file.

- **Added a warning when settings for current user are managed under a VirtualStore folder** due to lack of permissions on the shared AppData or ProgramData MilletSoftware folder.

- The dialog offering automatic conversion from mapped drive to UNC path now applies also to the email attachments option and to cases where Visual CUT is launched by double-clicking an rpt file in File Explorer.

- Added a **`Enable_Auto_Conversion_to_UNC`** ini option to allow **automatic conversion of rpt file, export file, and email attachment file paths from mapped to UNC convention**. When this option is set to True (default is False), mapped drive paths are automatically converted to UNC paths after a message box is displayed to the user.

- Added a `Display_Warning_Mapped_Drive` ini option to disable the dialog offering a conversion to UNC path when loading a report from a mapped drive.

- **Word_Replace_Tags** is now compatible with Word 2013.

- Added **Now_GMT_Plus_S** and **Now_GMT_Minus_S** to the date constant functionality. This allows setting parameters to values relative to the current universal (GMT) datetime.

Fixes

- VC 11 now logs a failure and closes when a scheduled report loses connectivity to the database in midstream (after a successful connection and while retrieving records). Also, when the report encounters a Division by zero and other unhandled exceptions. This avoids hanging instances of VC when such an event occurs. To disable this option, set `Detect_Lost_Connectivity=False` in DataLink_Viewer.ini

- Fixed a Connect_To_SQLOLEDB command line argument scenario that might cause a process to hang.

- Fixed a problem causing the Export Burst checkbox to lose its status after using the right-click report grid menu option of 'Copy .rpt and Settings'.

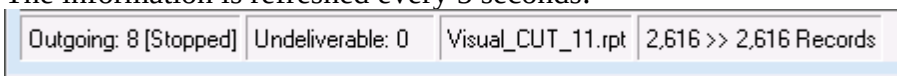
- Fixed a login issue for subreports when requesting a database server change.

- After_Burst_Batch, After_Export_Batch, and After_Success_Batch now generate temporary work files (when the called batch file contains references to fields/formulas) in the temp folder instead of in the Visual CUT main files folder. This avoids failures due to lack of permissions.

Version 6.7001: Released December 01, 2013

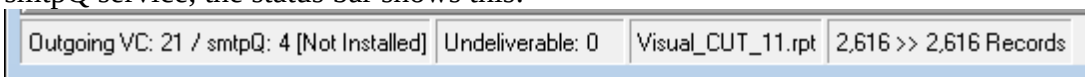
Email Features

- It is now easier to monitor the status of queued emails. If an Outgoing folder is specified, Visual CUT shows the number of email messages in the **Outgoing** and **Undeliverable** folders in the first 2 panels of the status bar.
  The panel showing the number of messages in the Outgoing Folder, also indicates if the smtpQ service is **Not Installed**, **Stopped**, or **Running**.
  The information is refreshed every 5 seconds:



- You can now open the **Outgoing** folder or the **Undeliverable** folder by simply double-clicking their status panel.

- If the Visual CUT Outgoing Folder doesn't match the Outgoing folder specified for the smtpQ service, the status bar shows this:



- Added a section describing a technique for "Slowing Down Outgoing Emails." This is useful when email queuing through the smtpQ service results in exceeding speed limits imposed by your email service provider.

- Added a user manual section about 'Embedding Hyperlinks to Reports/Files inside HTML Email Messages'. A sample report demonstrating this technique is available upon request.

- **Added a button to the Email HTML Editor to show dynamic field names in Notepad.**

- When queuing emails, `'` characters (as in `O'Brien`) in the .eml file names are now removed. This protects against emailing failures.

- Updated an email processing component.

- Improved handling of the optional **Email_SMTP_Domain** ini setting

- Added optional **Email_SMTP_Auth_Method** ini and command line argument option. In most cases, this setting is not needed since Visual CUT detects and automatically uses the most secure authentication method supported by the SMTP: "NTLM", "CRAM-MD5", "LOGIN", or "NONE".

- Updated the procedure for substituting accented characters in eml file names to handle additional Icelandic characters (þ, Þ, æ, Æ).

PDF Features

- Added **PDF_Split_By_Bookmarks** argument, allowing Visual CUT to split a pdf based on a targeted level of bookmarks.

- When using the **Print_Copies** argument to control the number of copies printed via the **PDF_Print** argument, **if the field or formula used to control the number of copies returns a zero, the printing is skipped**. This is useful in bursting scenarios where some recipients do not want a hard copy.

- Pdf processing tags are now removed even if they contain parentheses.

- Fixed an issue in PDF_Auto_File_Link when the text contains literal double-quotes.

Excel Features

- Visual CUT can now Replace or Append to an existing excel tab (using ***TabInOldFile!*** or ***TabInOldFile_Replace!*** directives) even when the tab is hidden.
  Also, fixed an issue with repositioning the workbook to its original selected tab.

- Added **XLS_Transfer_Tabs** command line argument, allowing tabs to be transferred and dynamically renamed to a new or existing workbook. This is particularly useful when bursting report information into named ranges (XLS_Range_Insert) in a template workbook, and then gathering the resulting tabs into a single workbook with multiple tabs.

- Added **XLS_Run_Macro** command line argument for triggering an Excel macro.

- **XLS_Range_Insert** now creates folders on the fly if the target file path doesn't exist.

- **XLS_Save_As** can now accept a tab name as a scope argument, so a specified tab name can be saved to a pdf file.

- **XLS_Save_As** can now save Excel files as HTML.

Web Dashboard & FTP/SFTP Features

- Expanded user manual section and added a **demo of a secure auto-refreshing web dashboard**. To access the web dashboard demo use demo as the user id & password.

- The **TXT_DeGUID_png** command line argument now handles cases where HTML exports reference multiple image files:
  1. Unique image files are renamed with incrementing index to differentiate them.
  2. Identical image files (e.g. repeating logos) are consolidated into a single image file.
  This avoids accumulating image files in web dashboard folders.

- Fixed an FTP_Upload issue when using Active_1 as mode.

- Added an ini file option (in the [Options] section) to disable EPSV mode when doing Passive_1 ftp uploads or downloads:
  FTP_Disable_EPSV_Mode=True
  This aims at fixing timeout problems when the ftp server mishandles EPSV mode.

- **SFTP_Upload** and **FTP_Upload** can now **skip files that are older than N minutes**. This allows specifying files via wild card expressions but targeting only new files.

User Interface Enhancements

- The text area showing the number of group values is now highlighted in red while Visual CUT is busy loading and counting these group values.

- The Option dialog has a new ***Database*** tab for setting connectivity options.

- **Ctrl-Shift-F1** now opens DataLink_Viewer.ini (configuration settings) in Notepad.

- When a user selects the option to convert a mapped path to UNC path for a report with saved settings, the process now checks to make sure the UNC path for that report doesn't already exist with saved settings.

- During interactive use, the Progress window now stays on top. This reminds users to close that window before clicking the Start button again.

- During forced login scenarios, Integrated Authentication now automatically sets the Password in the login dialog after the User ID is manually set.

Command Line Arguments

- Added **After_Success_Batch** command line argument supporting workflow automation and email notification after a successful completion of a Visual CUT process. Field/formula references within the batch file are dynamically replaced with their content from the processed report before the batch file is triggered.

- **Added support for specifying custom calendars as start or end points relative to date constants**. For example, this allows you to return the start or end of a the fiscal month relative to yesterday's date. For detail, see "***Custom Calendars***".

- Added a **Main_Files_Folder** command line argument for specifying the location of Visual CUT.mdb and DataLink_Viewer.ini. This is useful in scenarios where a centralized scheduler triggers processing on behalf of multiple users who maintain settings in their own folders.

- Added **Word Print** command line argument. Using fields/formulas you can dynamically specify the file to be printed, the printer, and the number of copies.

- Added a **Word_Protect** command line argument. This allows you to:
  a) **restrict viewing** of a document to only users who know the Open Password.

b) **restrict editing** of a document to only users knows the Modify Password..

- Added **TXT_Remove_Short_Lines** command line argument, allowing you to remove blank or short lines in a text file. This is useful for cases where TEXT exports generate blank lines or lines with just delimiters.

- **A new Set_Formulas1 command line argument allows setting of formula expressions** (provided the formula name starts with a ^ character). For detail, see the user manual section on *Using Command Line Argument to Set Formula Expressions*.

Fixes

- Fixed process logging when failure messages contain single quotes.

- Fixed double-space handling in command line arguments.

- Fixed a rare 'Row cannot be located for updating' error (when a report with linked dynamic parameter reports that don't match the connection properties of the main report) is loaded for the first time.

- Fixed tooltip display for dynamic field/formula values containing '&' symbol.

- Fixed handling of selective parameter refresh in an interactive use scenario.

- Fixed a scenario causing the export burst checkbox to lose its checked values during interactive use where the same report is refreshed or reloaded a 2nd time.

- When closing the application after an interactive session, the ReportList.txt file (1st tab report grid) is now transferred to the recycle bin approximately once per 10 times. This provides a copy of the file in case it gets erased during abnormal termination of a user session.

Other

- If a linked dynamic parameter report uses an ODBC DSN that doesn't exist, or if the **Set_Parameter_Rpt_to_Main_Rpt_DSN** option in the ini file is set to True, the DSN of the parameter report is automatically set to the DSN used by the main report.

- You can now direct the Visual CUT database to another DBMS such as SQL Server. This is particularly useful if you want multiple users to concurrently maintain Visual CUT settings in the same database. This is currently available only in Visual CUT 11. For detail, see ***Directing the Visual CUT database to Another DBMS***.

Version 6.6001: Released December 31, 2012

- **Visual CUT can now capture and process incoming emails**.
  This new features supports a variety of new use scenarios, such as:
    - Requesting a Report via a Simple Web Form
    - Updating a Database via a Simple Web Form
    - Collecting Customer Feedback & Updating a Database via Email Links
    - Importing data from email file attachments into a database table
    - Requesting & Capturing Management Decisions Via Email

  For more detail, see the new user manual section on **Capturing & Processing Incoming Emails**.

- You can now queue Visual CUT invocations so that at any point in time, no more than N instances are actively processing reports. For detail, see:
  "**Avoiding Too Many Active Visual CUT Instances (Queuing)**"

- **Fixed a problem with handling database connection failures during scheduled processing. This issue could cause Visual CUT instances to hang in memory.**

- Reduced/improved concurrent connections to the Visual CUT.mdb. This should also fix rare 'Error 91: Object variable or With block variable not set' problems.

- Fixed a problem caused by clicking on the Refresh button on the Preview tab. This could cause saved Burst option for the report to be disabled.

- **After_Success_SQL:**
  a) Added support for triggering multiple SQL statements (even for different ODBC DSNs)
  b) fixed processing when bursting and ODBC logging is disabled,
  c) if the SQL statement is blank, it gets skipped.

- Added two options to the Report Grid right-click menu:
    1. Open Containing Folder
    2. Open In Crystal

- Added **SFTP_Upload** command line argument supporting both regular as well as Private Key authentication.

- **Visual CUT can now handle cases where the command line arguments are separated by more than a single space.**

- Fixed a problem when connecting to OLE DB data sources.

- Updated several components to latest available versions.

- Double-clicking the ***Arguments*** field at the top of the Export/Email tab, or clicking a new button to the right of that field, now opens an **improved text editing window**. The window allows you to **increase/decrease font size**, and **switch between a Rich and Plain text format**. The Rich text format **highlights the names of acceptable command line arguments, as well as typical delimiters**. The window remembers its size and settings (font size, text mode).

- Fixed a problem with command line arguments containing '"^^"'

- The 'Change Stored Path Information' button (Process tab in Options dialog) now also takes care of updating file paths in the report grid (not just in Visual CUT.mdb). This is useful in cases where you need to change report locations or where you need to move Visual CUT to another server where the relative path to the report files is different.

- Fixed an issue with **Database_Path** ini file entry or command line argument when used to change the location of Excel files as a data source.

- Fixed an issue with reading very large ini file entries (saved parameter values).

- **Added warnings when an export file extension doesn't match the export format**. You can elect to turn off such future warnings.

- Added an option to suppress a warning message when changing ODBC DSN.

- The status bar now shows records **Read** >> **Selected** >> and **Group-Filtered**. Improved text message and handling of cases where an interactively loaded report results in zero Selected or Group-Filtered records.

- Added **Max Retries** option to the **smtpQ** administration window. This allows you to control how many attempts are made if an email queued for delivery fails to be sent. The delay before each subsequent retry keeps increasing (from 5 seconds for the 1st retry up to a maximum of 20 minutes for the 10th retry). The default number of attempts is 7.

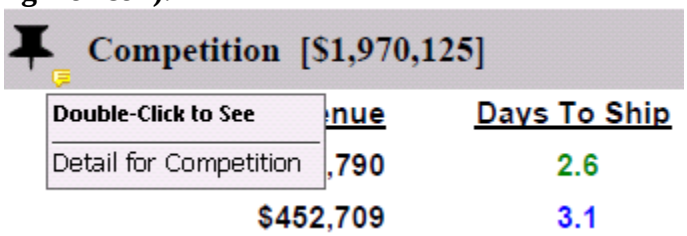
- From email address for failure alerts can be set via a **Email_Failure_Notices_From** entry in the [Options] section of DataLink_Viewer.ini.

- Added a user manual section about **generating Twitter or SMS messages**.

- Fixed a bookmark sort issue in PDF_Bookmark_Tags.
- Updated pdf processing component provides faster processing and ability to handle malformed pdf files.

- When adding page numbers to pdf files, you can now instruct Visual CUT to skip the last N pages. For detail, see '*Adding Page Numbers to a PDF File*'.

- When creating pdf drill-downs by embedding files using **PDF_Link_Tags2**, the Crystal tag formulas can now **specify the header line as well as the body line for the icon tooltip**. In the past, only the body line could be specified.
  Also the Crystal tag formulas can now **specify transparency level and type of icon (including No Icon)**.



- The **PDF_Properties** command line argument now supports specifying custom properties for the pdf document.

- Added an ini file option to specify a folder location for dynamic parameter reports.

- Added an ini file option allowing users to disable setting initial directory when browsing to open rpt files: **OpenFileDialog_Set_Initial_Directory**=FALSE
  This solved a problem for one user in a Citrix/TS environment.

- Fixed a problem with TXT_Replace when the replacing text is a Chr(n) expression.

- Added more detail to the ***Auto-Refreshing Web Dashboards*** user manual section.

- When the login dialog shows alternative ODBC DSNs to select from, if the machine has more than 9 DSNs, the dialog expands and allows you to enter text to **search the list of DSN**. This makes it **easier to locate a particular DSN in a long list of DSNs**.

- If you interactively switch between DSNs, Integrated Authentication functionality is now available by setting to **True** a new ini option of **Enable_Integrated_Authentication_For_DSN_Changes**.
  This is a rare scenario so, for more detail, contact Millet Software.

- You may now specify **ODBC_DSN_From_To** as a global entry in the [Options] section of DataLink_Viewer.ini. In cases where a global transition from old DSN to new DSN are desired, this removes the need to specify ODBC_DSN_From_To as a command line argument. The entry can have multiple pairs separated by "||".

- **Added a command line argument of Attempt_Logon_Without_Password**.
  If most of your reports use a data source requiring authentication, you can set the ini entry by that same name to False. For the few reports that use a data source without authentication, you can now override the ini setting by passing a command line argument such as …
  "Attempt_Logon_Without_Password:True".

- In cases where the report (or subreports) connect to more than 1 data source, Visual CUT 11 now supports a new ini [Options] entry of:
  Attempt_Logon_Without_Password_On_New_Server=True
  By default, that option is set to True, meaning that after successful connection to one data source, Visual CUT would attempt to connect to a 2nd data source within the same report using the user id & password established for the 1st data source. One customer encountered a problem when the 2nd data source is an ODBC connection to Progress. For such scenarios, users can now set this option to False.

- Depending on screen size, the selective parameter refresh grids now show more rows (reducing the need to scroll when a report has many parameters).

- Added Email_Message_Save command line argument to save the email message body to an HTML or Text file. This allows you to take advantage of the email message HTML editor to generate web pages with dynamic content from a Crystal report without using the HTML export.

- **Added a TXT_DeGUID_png** command line argument to remove GUIDs (unique identifiers) from png file references and file names in HTML exports. This can make the process of generating web dashboards faster, cleaner, and simpler. For detail, see "Removing GUIDs from png Files Referenced in HTML Exports."

- **Txt_Replace_Tokens** command line argument now has a 'DeleteLinkedFile' option designed for Web Dashboard generation cases where properly named linked image files already exist (and don't need to be changed). When using this option, after cleaning the HTML references to linked image files, the newly generated image files are simply deleted (instead of renamed). For detail, see 'Replacing Content in Text/HTML Files – Token Approach.'

- If the **Email_To** option contains the text "**VC_Skip_Email**", the actual emailing of the message is skipped. This allows you to use a formula that returns an email address or "VC_Skip_Email" to control whether an email will be sent out. It also allows you to use the new **Email_Massage_Save** command line argument without emailing anything.

- When queuing emails as .eml files to an outgoing folder, if the Email_From option contains the text "[VC_Blank_Address]" the email from option is set as blank. This supports rare use scenarios where the user wishes to open the .eml files in Outlook and populate the From address (and the certificate) based on the user's Outlook settings.

- When queuing emails, accented characters in the .eml file names are now replaced with unaccented characters. This protects against emailing failures.

- **XLS_AutoFit** now provides arguments for **specifying a maximum column width and whether or not the content of maxed-out columns should wrap**. For detail, see "Column Auto Fit in Excel Exports"

- **XLS_AutoFilter** now provides two new arguments. The first argument specifies the **cell location where AutoFilter should be applied.** This is useful when the tabular range doesn't start at the top of the spreadsheet. The second argument specifies a cell location where **Freeze Panes should be applied**. For detail, see "Auto Filter & Freeze Panes in Excel Exports"

- **XLS_Range_Insert** now takes care of setting the resulting workbook to open to the original sheet and selection that were last set in the source workbook.

- **Macro-Enabled Excel workbooks (.xlsm) can now be targets for adding or replacing tabs** (***TabInOldFile*** or ***TabInOldFile_Replace*** functionality). This allows your workbook to apply Macro logic to the refreshed data.

- Fixed a problem with specifying RGB color for excel tabs.

- The **Print_Copies command line argument can now handle field/formula references**. For example, "Print_Copies:{@Label_Quantity}"

- **XLS_Pivot_Table** can now handle cases where rows or columns have extremely large number of unique values (provided your machine has Excel 2010 installed).

- ODBC export with REPLACE option now deletes records (rather than DROPs the table) before the export. If the table has an index, this ensures the index is preserved.

Version 6.5001: Released January 26, 2012

- Added **After_Success_SQL** command line argument, allowing you to **update a database after each successful bursting step or full report**. For example, after bursting invoices to customers, you can update the records for these invoices to reflect the date of invoicing or the fact that an invoice was emailed. For more detail, see "Update a Database After Success (After_Success_SQL)."

- **The old email engine has been deactivated and the 2009 email engine is now always used**. If your installation used the old engine, you will receive a detailed message about the changeover when you first start the new Visual CUT version.

- **Each time Visual CUT starts it now alerts you if there are new email messages in the Undeliverable folder of the smtpQ service**. If you specified an email address in the Log/Alert options dialog, an email goes out to the specified address. Also, if Visual CUT started interactively (not via a command line), a message box is displayed. **This provides early detection of situations where the smtpQ service was unable to send email messages**.

- **Visual CUT now takes care of replacing illegal characters and removing non-printing in export file names, email file attachments, and .eml files**. For more detail, see: "Replacing Illegal Characters in Dynamic File Names."

- **If you specify .xlsx file extension for Excel 97 exports, you now get an Excel 2007 (.xlsx) file format, supporting more than a million rows in a single tab and consuming much less disk space.** For more detail, see "Exporting to Excel 2007 (.xlsx) Files." Note that this functionality is **particularly useful when exporting a large report to Excel as a basis for automatically generating a Pivot Table** (XLS_Pivot_Table argument).

- **XLS_to_XLSX** command line argument was added to support file conversion needs beyond direct exports.

- **XLS_Save_As** command line argument was added to support converting excel files to pdf files.

- **XLS_Pivot_Table** has a new optional argument for **setting the orientation of multiple data elements to columns instead of rows**.

- **XLS_Pivot_Table** now handles cases where specified metric caption is identical to the column name.

- **XLS_Pivot_Table** error messages now provide more specific information to help identify the text element causing the error.

- Excel exports with **Tab!**, **TabInOldFile!**, **TabInOldFile_Replace!**, and **TabInNewFile!** can now specify **.xlsx** workbook destinations.

- Excel workbooks generated with **TabInOldFile!** or **TabInOldFile_Replace!** processes now **open with the first tab in the workbook as the default tab**.

- **Visual CUT no longer logs processing for bursting steps that are skipped due to Skip_Recent**. This speeds up processing and keeps the log table smaller.

- If the export file name is specified as **VC_Skip_Export** (no file extension and no file path) Visual CUT now skips the actual exporting step, saving time in cases where the export is used as an excuse for other special Visual CUT processing.

- **Failure log entries now start with the path and name of the failing report**. Also, when a bursting step fails, the **failure log entry and email alert now include the Group Level 1 Value information**.

- You can now specify (in the DataLink_Viewer.ini) a list of **DateTime parameter names that should be set by date constants to the <u>end</u> of the day** (11:59:59 PM). For detail, see the Date Constants section in the user manual.

- **ODBC_DSN_From_To now support multiple pairs** separated by "||".
  This addresses use scenarios where a report uses multiple ODBC DSNs (e.g. when the main report DSN is not the same as subreports' DSNs).

- **Added Database_Path command line argument (and similar ini entry) to change path to native Access, Excel, and Pervasive (ddf) data sources**. For detail, see "*Changing Folder Location for Access/Excel/Pervasive (ddf) Files*"

- Added a new **Txt_Replace_Tokens** command line argument to process Text or HTML file exports. A typical use scenario is HTML exports where image files (logos, charts) are generated with random names. This allows you to:
  a) **replace file names in img src HTML tags with specified file names**.
  b) optionally, **rename linked image files**
  For example, when using Visual CUT to generate web dashboards (see 'Auto-Refreshing Web Dashboards' section), this new functionality removes the need for uploading multiple image files with random file names. It can also improve browsing performance through file caching.
  For detail, see 'Replacing Content in Text/HTML Files – Token Approach.'

- Online Version Update functionality was removed due to the component Update.exe being (mistakenly) flagged by virus protection software.

- **You can now automate the process of updating entries in DataLink_Viewer.ini by placing an update file in the application folder**. For detail, see "Updating DataLink_Viewer.ini via a Delta File."

- Ctrl-A now select all content in text fields (in export/email tab).

- Added automatic removal of leading spaces from options such as export file name, email addresses, etc. Such leading spaces could cause processing failures.

- Fixed an issue caused by record selection formula containing only a comment.

- In the report grid, disabled delete action for a group to prevent accidental removal of all report rows belonging to that group.

- When changing the grouping of the report grid, if the grid contains more than 100 rows, instead of expanding all group nodes, all group nodes are collapsed.

- Fixed a problem when a legacy user who hasn't added yet the "Arguments" column to the Report_Opt table in Visual CUT.mdb tries to use the menu option to copy report settings from one report to another report.

- Added **Write_INI_Location** command line argument (to write the path to DataLink_Viewer.ini) to a specified text file.

- Fixed a connectivity issue (may manifest as "Type mismatch" error) to the Visual CUT.mdb on machines with operating systems earlier than Windows 7 SP 1.

- When exiting the application or moving away from the 3rd (email/export) tab after making changes to report processing options, a message box allows you to elect to save or discard those changes.

- Fixed a problem with command line processing when the report has no saved settings.

- Fixed a problem in some situations when opening the Options dialog.

- Fixed a problem with Before_Export_Batch.

- Fixed a problem with specifying a list of printer destinations in a text file.

- Fixed a recently introduced bug causing failure when bursting very large reports.

- The HTML editor, Visual CUT now **automatically removes false email links** generated while editing a line containing a formula references (due to the '@' symbol). When a link is removed, the cursor moves to the end of the message.

- **The Preview window in the email HTML editor now shows the dynamic values of email *from, to, cc, bcc, attachments,* and *subject***.

- If all email destination options (to, cc, bcc) are blank, a failure is triggered to avoid placing such email messages in the outgoing queue.

- Fixed a rare email queuing problem (email addresses that start with 'con.').

- If an export fails due to a long file path and name (> 260 characters), the failure message now indicates this may be the reason for the failure.

- Added a Word_Replace_Format command line argument. This functionality was developed to allow an export from Crystal to MS Word to result in Double Underline (typical for totals in accounting reports).

- Updated the *Known Issues and Limitations* user manual section discussing how to improve image resolution when exporting to PDF/RTF/Word. The instructions and sample .reg files now cover Visual CUT 11 as well.

- Fixed a bug in FTP_Upload causing successful wild-card uploads to report a failure.

- Added more detailed progress dialog information to several pdf processes.

- Updated pdf processing component.

- Links to pdf files generated using **PDF_Auto_File_Link** or **PDF_Auto_File_Link_Tokens** now open linked pdf files using their default zoom and page layout settings and in a new Window.

- Fixed a problem for some pdf files when using **PDF_Form** command line argument.

- When using **PDF_Form_Tags** to generate form fields, **you can generate *child* fields** by specifying the field title as ***parent_field_title.child_field_title***. This is useful for submitting form data as XML hierarchy.

- **PDF_From_TIFF** now supports merging multiple TIFF files into a PDF file.

- Added **PDF_Add_Media** command line argument allowing you to embed a sound file for auto-play when opening the pdf file. For detail, see "***Adding Multimedia to PDF Document***."

- Added **PDF_Build_Index** command line argument. For detail, see "***Building an Index PDF Documents.***"

- **PDF_Properties now supports setting properties for multiple files** specified as a comma delimited list. The files can also be specified using **wild cards**.

- Fixed a problem with PDF_Bookmark_Tags (introduced by the new pdf component).

- Fixed a problem with **PDF_Protect** when processed pdf file is not the exported file and not all optional arguments are provided.

- Fixed a problem in PDF_Link_Tags.

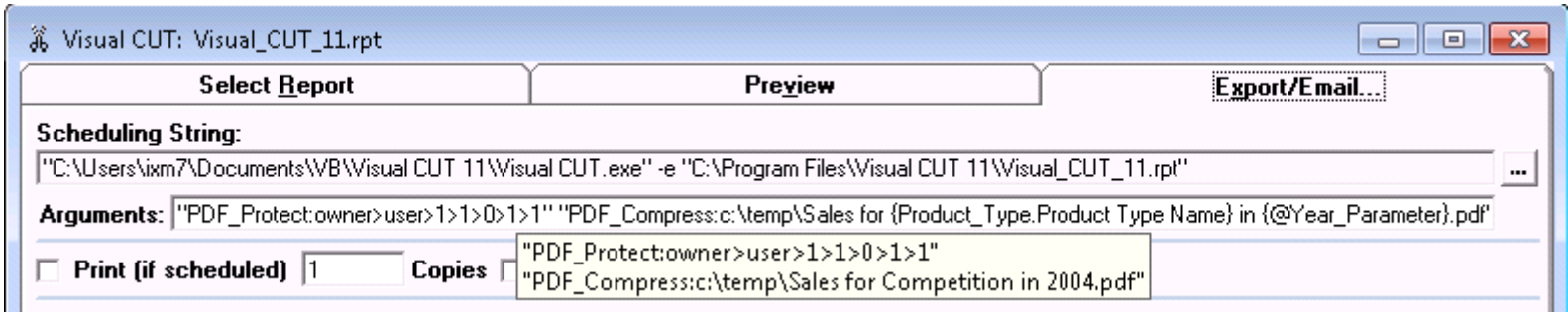
- **PDF_Protect** now checks to make sure the owner password is not blank.

- **PDF_Protect now allows you specify encryption strength of 256-bit AES**.

Version 6.4001: Released June 01, 2011

- **Visual CUT can now generate Excel Pivot Tables**. For detail, see "Generating Excel Pivot Tables" " or watch a **video demo**.

- **You can now save and apply command line arguments during interactive use**.
  A new ***Arguments*** field in the 3rd (export/email) tab (and in the Visual CUT.mdb) allows you to specify and save arguments for use during **interactive** use (click Start Process button) as well as during **unattended** command line processing.



  A double-click on this field opens the text in a multi-line editing mode.
  Command line processing honors these saved arguments but can override them.
  Users of prior versions who wish to take advantage of this new feature should add an **Arguments** column (**Memo** data type**)** to the **Report_Opt** table in **Visual CUT.mdb**
  For more detail, see "Specifying Processing Arguments through the GUI"

- **When a user interactively loads a report that has saved settings, Visual CUT now provides a dialog that allows the user to reuse all or some of the saved parameter values**. This matches typical use scenarios and removes the need to enter all parameter values each time a user opens the report interactively.

- For situations where reports with many parameters are used in one of several parameter patterns, **you can now save & reuse Named Parameter Sets**. For detail, see "Save and Reuse Named Parameter Sets" or watch a **video demo**.

- **Fixed an issue that can cause instances of Visual CUT to remain in memory**.

- **Added After_Export_Batch and Before_Export_Batch command line arguments**. Like **After_Burst_Batch**, this allows you to not only execute batch files but also temporarily replace fields/formula references inside the batch file with their dynamic values from the report. For example, you can archive exported files into a .RAR format before emailing or FTPing. For more detail, see: ***Triggering a Batch File with Dynamic Content Before/After Export*** .

- Added a user manual section on how to create "***Auto-Refreshing Web Dashboards***."

- Fixed print queue job renaming during bursting.

- Added a **Release_Shared_File_Before_Export** option in DataLink_Viewer.ini. You can turn this on through the Options dialog if you wish to always attempt to **unlock the exported file from shared use by network users who may have a prior version of the file opened in a shared network destination folder**. If you wish to override the default behavior controlled by that option, you can use a new **Release_Shared_File** command line argument. For example, if the global option is set to False and you wish to enable it for a specific report, add … **"Release_Shared_File:True"** to the command line.

- **Word_Replace_Tags** now supports dynamic references to fields and formulas.
  It also opens the template document in Read-Only mode. Also, an issue with Append functionality in Word_Replace_Tags has been fixed.

- Added **Word_Save_As** command line argument. **This allows you to convert Word documents to PDF documents**. Among other features, **this allows you to generate PDF/A documents and to handle special fonts that can't be handled via direct PDF exports** (by first exporting to Word, and then converting to PDF).

- Both **Word_Replace_Tags** as well as **Word_Save_As** now get processed before any PDF processing command line arguments. This allows you to convert a Word document to PDF and perform extra processing on the resulting pdf file within a single command line.

- **Added PDF_Link_Tags2 command line argument. This allows you to use Crystal formulas as hidden tags that control not only the embedding of files inside a PDF file but also the creation of visible links to these embedded files. This allows you to deliver pdf files that act as containers for drill-down or other related content.** For detail, see "Adding Links and Embedded Files Using Crystal Formulas as Tags" or watch a **video demo**.
  For a sample PDF with embedded files for drill-down functionality, see:
  http://www.milletsoftware.com/Download/Visual_CUT_PDF_Embedded_Drill_Down_Sample.pdf

- **Added PDF_Embed command line argument. This allows you to embed multiple files as attachments inside a main pdf file**. For detail, see "Embedding Files as Attachments inside a PDF File."

- Updated pdf processing component, providing faster processing and better handling of some pdf files that were created by other sources in a malformed manner.

- The Form Field **Text** and **Description** options in **PDF_Form_Tags** now support SQL Query directives. This allows you to stay within the 1 line limitation for the tag but still specify very long text for situations such as long comments. For detail, see "Populating Form Field Text or Description via ODBC Query". Also, specifying "Same_As_Text" for the Field Description option sets the tooltip as the Field text.

- Several fixes and enhancements to **PDF_Auto_File_Link**. One key enhancement is the ability to target text in the pdf file and generate file links for it based on wild card tokens. For a detailed description, see "Detecting Additional File References Using Wild Card Tokens."

- **PDF_LinkTag** now handles cases where the tag contains spaces after the closing '#'.
- Fixed an issue related to detection of text position inside pdf files.
- Added file compression to some pdf processes (reduce the need for PDF_Compress).
- Fixed color handling issue with pdf Bookmark Tags.
- Added a **PDF_Split_Protect_Tags** command line argument. This is like **PDF_Split_Tags** but also adds the ability to protect the split pdf files.
  For detail, see "**Splitting and Protecting PDF Files**".
- Several speed enhancements. Also, PDF_Split_Tags, PDF_Split_Protect_Tags, and setting pdf document properties now consume less memory.
- **PDF_Print** now support two optional argument for a) controlling fitting the printout to the page size and b) automatically rotating and centering the printout.
- **PDF_TOC** now supports a **Page_Orientation** directive (Poertrait or Landscape). For more detail, see *Advanced Table of Content Options*.
- **PDF_Merge** can now **generate multi-level bookmarks based on the file names of the merged files**. For more detail, see the user manual section on "Using the Merged File Names to Generate Multi-Level Bookmarks."
- Fixed a PDF_Merge issue with the old approach (no method argument).
- PDF_From_TIFF now automatically matches the TIFF page orientation.
- **PDF_Protect** can now create a new protected file and leave the original file as-is.
- Setting the PDF_Bookmark_Tags_Default option to True (in the ini file) now applies only to PDF files that are smaller than 100MB. This avoids wasted time and rare "out of memory" issues.
- **Double-click on the export file name opens the folder location in File Explorer**. This makes it easy to navigate to the export file location after processing is done.

- In the Version Information dialog, a double-click on the bottom text area showing paths to main files (DataLink_Viewer.ini, Visual CUT.mdb, ReportList.txt), opens that folder in File Explorer. This is particularly useful when the folder is hidden. The dialog also shows the 'Build Date' of the application.

- Fixed an issue with the report list grid not remembering expanded/collapsed state of some report groups.

- Fixed a problem with command line exporting to rpz file format.

- Fixed an issue with using date constants in multi-value range parameters.

- Fixed an issue causing an attempt to set non-string parameters to an empty string when a parameter has no saved value (in Visual CUT.mdb).

- Fixed an issue with specifying a simple Parm8 argument when the saved value for Parm8 (in Visual CUT.mdb) is a composite of values for multiple parameters.

- Fixed an issue with the 'copy .rpt and settings' process.

- Job status files now defaults to the location of Main_Files_Folder. Also, added logic to handle multiple Visual CUT instances updating the same job status file.

- During command line processing, connections to the Visual CUT.mdb database are closed more quickly, reducing the chance for conflict between batch files.

- Skip_Recent scenarios no longer log the event to Failure.Log. If you wish to log such events, set **Log_Skip_Recent_In_Failure_Log** to True under the [Options] section of DataLink_Viewer.ini .

- Fixed a bursting issue when selection formula contains strings such as: http://...

- Added hours {[hh]}, minutes {[nn]}, and seconds {[ss]} to the list of fields/formulas for drag and drop.

- When clicking the User Manual button, the user now gets a message about the need to install a pdf file reader if the machine doesn't have an application associated with the .pdf file extension.

- The browse button for selecting a text file to act as an Email To distribution list now remembers the last used folder location.

- If you add the following line to the [Options] section in DataLink_Viewer.ini:
  Email_Warning_On_Folder_Creation=True
  Visual CUT would email the address set in Log/Alert tab a notification each time it creates a new folder (for example, in cases where the export folder doesn't yet exist).

- Several GUI enhancements to the 'email 2' options tab:
  a) If a user without administrator rights click on the Administer smtpQ button, a message box explains that administering the service may require that the user be logged in with administrator rights.
  b) The status of the smtpQ service (Installed/Not installed, Running/Stopped) is now visible (for users with Administrator rights) without needing to go into the Administer smtpQ Service dialog. Double-clicking these status indicators change the status. For example, if the smtpQ service is stopped, a double-click would start it.
  c) Attempts to use a network drive for the Outgoing Folder are blocked with a message box explaining the smtpQ folders must be on the local hard drive.
  d) If the Outgoing Folder option is blank, a double-click sets it to the default of: "C:\Visual CUT\smtpQ\Outgoing"

- Added support for relative XML file paths in tables used within RPZ reports.

- On first launch of Visual CUT after a 1st-time install, if the main settings folder gets relocated to a common application data subfolder, Visual CUT provides a detailed message about the need to set permissions for that folder. Otherwise, under Vista and Windows 7, different versions of settings might be maintained for each user in individualized \VirtualStore\ folders. To facilitate changing subfolder permissions, the message provides an option to automatically open it in File Explorer.

- Added options for disabling buttons and editing options within Visual CUT. For detail, see "Restricting User Actions."

- The batch file creation dialog now allows selection and creation of command files (*.cmd) in addition to the batch files (*.bat).

- Suppressed the large tooltip for email message bodies if they are HTML. Such messages are better previewed via the email HTML editor.

- Added a DataLink_Viewer.ini option to automatically close a dialog (Outlook not being default email client) during exporting. To enable, set this option to True:
  Auto_Close_Dialogs_During_Export=True

- The email HTML editor can now handle cases where web links and file links within the email message contain embedded references to Crystal fields and formulas.

- Progress Window now shows more detail and clear separation between bursting steps.

- **Fields and Formulas placed in Report Headers and Footers are now recognized not only in the first section but also in sections b,c,d, and e**. This allows you to place a subreport in a higher section (for example in RFa) and use a shared variable formula in a lower section (for example in RFb) to get a value from the subreport and pass it to Visual CUT.

- The email HTML editor now allows you to move to previous/next groups while previewing the message. This allows you to see how the message changes when bursting to different groups.

Version 6.3001: Released July 02, 2010

- **Visual CUT can now save any number of parameters, including subreport parameters.** In prior versions, only the first 8 parameters from the main report were saved. For command line processing, parameters 9 and above had to be specified via command line arguments. Also, in prior versions, independent (unlinked) subreport parameters had to be changed into linked parameters. Note: no change to your Visual CUT.mdb tables structure is needed (all the extra information is stored inside the existing Parm8 column).

- **Visual CUT can now FTP files to a web server**. This makes it easier to provide access to reports via web or email links. For detail, see "Exporting/Uploading to an FTP Server (new approach)"

- **Added a powerful HTML editor for composing email messages**. For detail, see the user manual section on "***Integrated HTML Editor***".

- Added a video demonstration of embedding dynamic HTML tables with report information inside an email message body, and using the HTML email editor.

- Added functionality and instructions for embedding images in HTML email messages when using the email 2009 engine. **Inline images are now embedded using an approach that is better supported by most email clients**. When using the new HTML editor, all you need to do is insert the image. For detail, see "Embedding Images inside the HTML email body (2009 email engine)."

- You can now **control the color of tabs when exporting to excel**. For detail, see: "Controlling Excel Tab Colors."

- Enhanced multi-line tooltips for displaying dynamic content of export/email options.

- In the Select Reports Tab, you can now use **Ctrl-F** to search for a row in the report list containing specified text. **F3** searches for the next matching row, and **Shift-F3** searches for a previous matching row.

- Fixed scheduling string handling with report names that contain single quotes.

- Fixed alert handling of cases where Printer name is not valid.

- Added a detailed message when a user tries to save report processing options to a write-protected visual CUT.mdb file or folder.

- When using wildcards for email file attachments, you can now elect to recursively include matching files in lower-level folders by setting the following entry in DataLink_Viewer.ini:
  [Options]
  File_Attach_WildCard_SubFolders=TRUE
  This is typically useful in cases where email attachments need to adjust to the recipient's level in an organizational hierarchy.

- You can now instruct Visual CUT to avoid registering iSED.exe on each load of the application by setting the following entry in the [Options] section of DataLink_Viewer.ini to False:
  [Options]
  Register_iSED_On_Load=False

- During command line processing, connections to the Visual CUT.mdb database are now set to be Read Only. This should **reduce the chance for a rare locking scenario when multiple reports are processed at the same time**.

- Visual CUT now generates a failure message when a command line argument is not recognized. This allows **detection of misspelled command line arguments**.

- Visual CUT XI now uses a newer version (Service Pack 6) of the Crystal XI R2 runtime components. Users of prior XI versions can't use the online update option and must go through a Remove & Install process in order to update their version.
  Note: this update **brings back the missing Cancel button in parameter dialogs**.

- Updated runtime components related to emailing, file compression and encryption.

- Double-clicking the list of Fields & Formula names area in the Export/Email tab, now opens that list in notepad for easy copy & paste into command line arguments and template files.

- You can now use the Graphical User Interface to set settings for and trigger Word_Replace_Tags processes. Existing users should add a new row to the **Export_Opt** table in the **Visual CUT.mdb** Access database:

Export Constant	Export Name
Word_Replace_Tags	Word_Replace_Tags

  Also, open the **Report_Export_Options** table and add a field called **Extra_Options** with a **memo** Data type.
  Note: this new functionality is not available in the 8.5 version of Visual CUT.

- Optimized handling of strings in several procedures for greater speed.

- Fixed an issue with the Burst Export checkbox showing an incorrect value when switching between reports.

- On first-time start of Visual CUT or if the application folder is write-protected (typical with Vista or Windows 7), Visual CUT creates an application settings folder, such as:
  \MilletSoftware\VC_11\
  in the common application settings directory (e.g., c:\ProgramData) and copies/redirects the locations of Visual CUT.mdb, DataLink_Viewer.ini, ReportList.txt, and ReportList.grd to that folder. The user is notified when the initial redirect occurs. **This removes the need for users to manually redirect the location of these files or to manually add Write permissions to the program folder**.

- The version information window now provides the current file paths to the Visual CUT.mdb, DataLink_Viewer.ini and ReportList.txt files.

- Added YearMonth_AT_PLUS_MONTHS_N and YearMonth_AT_MINUS_MONTHS_N number constant to support cases where users control date ranges with numeric parameters structured as YYYYMM.

- Added a **Use_Saved_Date** command line argument. This is useful in cases where you need to override the global setting in the ini file. For example, if the global option of Use_Saved_Data_in_Scheduled_Reports is set to False, you can add … **"Use_Saved_Data:True"** to a command line for a specific report.

- **Added handling of rpz files with embedded expiration date and/or license key requirement** (new features provided by DataLink Viewer).

- The Options Dialog (Process tab) now provides a **button for initiating batch updates to report paths** in 3 key tables within Visual CUT.mdb (Rport_Opt, Login_Opt, and Report_Export_Options). This helps with situations where the location of the rpt files must be changed.

New PDF Features

- A new component for pdf processing provides **faster processing**.

- **New PDF_Form_Tags processing options**:
    1. add **Checkbox** fieds
    2. add **DropDown** fields
    3. add **Submit button** (automatically send form data as XML via email or url)
    4. designate form fields as **hidden**
    5. designate form fields as **Read Only**
    6. designate form fields as **mandatory**
    7. designate form fields as **mandatory depending on another field value**
    8. Visual CUT automatically adds **JavaScript code** to the pdf to **turn background color of missing mandatory field values red**. Also, when mandatory field values are missing, attempts to submit the form data are blocked and the user gets a message box listing the missing form field values.

    Sample: **www.milletsoftware.com/Download/Visual_CUT_PDF_Export_with_Form_Fields.pdf**
    See "Adding Form Fields to PDF Files" for more detail.

- Added a **PDF_Sign** command line argument. For detail, see "Adding Digital Signature to a PDF File."

- PDF protection via the PDF_Protect command line argument now uses **advanced 128-bit AES encryption**. Also, 3 more optional protection directives were added (allow form field filling, allow copy for accessibility, and allow document assembly). Acrobat Reader 7.0 or later is needed to open the resulting pdf file.

- **Visual CUT can now remove pdf processing tags from the pdf file after processing those tags**. The main benefit is that text searches and indexing no longer include those tags. It also means you no longer need to set the background color of these tags to make them invisible. This functionality is controlled by an entry called **PDF_Tags_Delete_Default** under the [Options] section of DataLink_Viewer.ini.
  By default, this option is set to True.
  Note: to ensure all tags are removed, use the **Replace()** function in Crystal to change double quotes into single quotes.
  For example, the expression for the title in PDF_Bookmark_Tags could be:
  Replace({Customer.Customer Name},"""","'")

- Added a **PDF_Bookmark_Tags_Default** option to the [Options] section in DataLink_Viewer.ini. If you set it to True, Visual CUT would automatically attempt to generate bookmarks based on tag formulas inside an exported pdf. **This avoids the need to trigger the report from a command line with a PDF_Bookmark_Tags argument**. Note: using a PDF_Bookmark_Tags command line argument overrides the ini file setting.

- Fixed a rare case were order of bookmarks generated by PDF_Bookmarks_Tags could be wrong.

- Fixed a ‘Subscript out of range’ issue in PDF_Link_Tags.

- Fixed ‘Type mismatch’ errors in pdf tag processing on machines where the regional setting for a **decimal separator** is not a dot. For example, in French Canada where it is a comma.

- Fixed a problem with pdf Table of Contents page targets.

- PDF Table of Contents are now generated after the new PDF_Bookmark_Tags processing. This allows both bookmarks as well as Table of Contents to be generated in a single command line call.

- Added two more alternative processing modes (M2 and M3) to PDF_Merge. These options use a newer pdf handling component.

- The **PDF_Properties** command line argument now **allows setting of initial viewing properties: visible panel (outline/bookmarks, thumbnails, full screen, optional content group, attachments), initial page number, and zoom**.

- Added a **PDF_Compress** command line argument to reduce the size of PDF files (particularly after PDF_Merge operations). For detail, see "Compress PDF Files".

- Added a **PDF_Auto_File_Link** command line argument, allowing Visual CUT to **automatically detect file references inside pdf text and automatically add colored hotspots over these file references that link to the specified files**.

- Added a **PDF_Split_Tags** command line argument providing **amazingly fast bursting of very large reports**.

Version 6.2001: Released November 28, 2009

- Added **PDF_Bookmark_Tags** command line argument allowing you to generate bookmarks in pdf files based on invisible Crystal formulas acting as tags. For detail see the section on "Adding Bookmarks Using Crystal Formulas as Tags [New Approach]." Compared to the old approach, this new technique has several advantages:
    1. The bookmarks links to the **exact vertical location** (on the linked page) where the rendered formula is located. In the old approach, bookmarks only pointed at the top of the page.
    2. You can easily **generate bookmarks from within subreport or from any report section** (not just Group Headers).
    3. For each bookmark node, you can **dynamically control (using formula logic):**
        a. **the text color**
        b. **the expanded/collapsed state**
        c. **the style** (Regular/Bold/Italics)
    4. No need to worry about Keep Together and WhilePrintingRecords properties that can cause headaches with the old approach.

- Added a **PDF_Link_Tags** command line argument, **allowing invisible Crystal formulas (acting as tags) to direct Visual CUT to add file/page links and images to pdf exports.**

- **Added advanced options for generating Table of Contents in exported pdf files. These option allow you to insert web-linked image headers and control font type, color, size, spacing, indentation, and bullets for each level in the table of contents**. A sample pdf file is available here. For detail, see "Advanced Table of Content Options".

- Visual CUT now **maintains bookmark colors during merging of pdf files**.

- Added **PDF_Insert_BackPage** command line argument, to support scenarios were static content has to be added to the back of each page before printing the report.

- Added **PDF_Add_Index** command line argument to allow specifying an index file associated with the pdf file.

- The **PDF_ Page_N** command line argument can now specify any **True Type font and color**. For detail, see "Adding Page Numbers to a PDF File."

- The "PDF_Merge:List_File: …" argument now handles cases where the text file specifying the files to be merged contains dynamic references to fields/formulas.

- Added a 4th optional "method" argument to PDF_Merge. An alternative merge method solves a rare merging issue when using newer pdf files with form fields. For detail, see "Merging PDF Files".

- PDF_Merge operations can now handle cases where file names contain a comma.

- Fixed a problem related to generating pdf bookmarks.

- Added **XLS_Range_Insert** command line argument to **replace named ranges inside MS Excel spreadsheets with values of Crystal formulas (providing values for a single cell, single row, or even a tabular range)** with matching names. The file can then be saved to a new dynamically named file and emailed as part of a bursting Visual CUT process. The advantage of this technique is that you can keep and control all elements in the template spreadsheet. For example, the template spreadsheet may have formulas, conditional formatting, hidden columns and macros. For detail, see "Inserting Crystal Values into Excel Templates."

- Added **XLS_Protect_Worksheets** command line argument to **allow protection of some spreadsheet content from user viewing/editing**. For detail, see: "Protecting Excel Worksheets against User Viewing/Editing."

- Added support for exporting to a protected (.rpz) report file. This allows you to schedule exporting to protected Crystal Report files with saved data. The intended user will be using DataLink Viewer to open and interact with the .rpz file just like an .rpt file (except they will not be able to see or change how the report was designed). Existing users should add two new rows to the **Export_Opt** table in the **Visual CUT.mdb** Access database. The 2nd row is for exporting to rpz files for use by DataLink Viewer 2008 (SP2 or higher):

Export Constant	Export Name
Protected Report (.rpz)	Protected Report (.rpz)
Protected Report 2 (.rpz)	Protected Report 2 (.rpz)

- Improved default settings for command line exporting of reports with no saved settings for "Excel 97 (Data only)" format to avoid blank columns (in VC 11) and for "TEXT" format to avoid pagination (in VC 9 & 11).

- **Added warnings when an export file name was specified without a path or without an extension**. The user can elect to turn off such future warnings.

- Email alerts about processing failures are now using the new email engine (Email 2009) if that email engine was enabled.

- **Added a warning when doing a <u>whole</u> export and attaching the file to an email <u>burst</u>** (perhaps the user forgot to turn on the export bursting option). You may turn off these warnings from the Options dialog.

- The **following command line arguments now allow blank values**:
  EXPORT_MODE (to turn off exporting), EMAIL_MODE (to turn off emailing), EMAIL_TO, EMAIL_REPLY_TO, EMAIL_CC, EMAIL_BCC, EMAIL_ATTACH, EMAIL_MESSAGE.
  This allows users to turn off (set to blank) these options in cases when the option has a saved setting for that report.

- **Mapped Drive → UNC Path Functionality:**
  1. Added a warning when a report (selected from the Open Folder button) or an export file path uses a mapped drive. The user can elect to automatically replace the mapped drive path with the equivalent UNC path (\\ServerName\Share\…). This avoids failures during scheduled processing when the mapped drive is not recognized by the Local SYSTEM account.

  2. Right-clicking a report row in Select Reports grid provides a new option (visible only for reports that use a mapped drive path) for converting mapped drive paths to UNC (\\ServerName\Share\…) paths. The conversion occurs in the grid itself as well as for the saved report path settings in Visual CUT.mdb. Using UNC instead of mapped drive paths avoids failures during scheduled processing when the mapped drive is not recognized by the Local SYSTEM account.

- Logging processing to an ODBC table now handles cases where the export file name contains single quote (‘) characters. Also, cases where connecting to the ODBC target fails are now handled more smoothly.

- Email alerts about processing failures are now using the new email engine (Email 2009) if that email engine was enabled.

- Fixed an issue with Emails with missing attachments.

- Brought back the Refresh button in the Preview tab for Visual CUT 9.

- **Added support for emailing via a SOCKS proxy**. DataLink_Viewer.ini now allows you to specify all the necessary options: Email_SocksHostname, Email_SocksPort, Email_SocksUsername, Email_SocksPassword (supported only when using SOCKS 5), Email_SocksVersion (4 or 5). This functionality is supported only when using the new Email 2009 engine.

- Added an **Email_SMTP_Disconnect_After_Send** option (in DataLink_Viewer.ini). This options applies only when using the new Email 2009 engine. By default, after an email message is sent, the new email engine disconnects from the SMTP server. This option, when set to FALSE, allows you to avoid disconnecting.

- Fixed a problem with logging processing to MS_Log table when a command line contains single quotes.

- Fixed two issues related to ODBC exports (one related to MS Access .accdb databases files, and one related to timeouts when processing huge files).

- Fixed an issue related to using Skip_Recent and multiple Printer_Burst destinations.

- Fixed handling of empty strings in both stored parameter values as well as in command line arguments (for example, …`"Parm1:"`).

- Printer_Burst and Printer_Burst_Only can now **specify multiple printer destinations using a text file listing the printers**. For detail, see "Printing to Multiple Printers."

- **Added code to silently log special data source failures during command line processing. This should address rare cases, such as database deadlock situations, causing scheduled processing to hang**.

- Visual CUT now **allows emailing when no Email_To is specified**. This accommodates scenarios where **Bcc** is specified without Email_To.

- Enhanced handling of command line processing with switched ODBC DSN data sources (using **ODBC_DSN** or **ODBC_DSN_From_To** argument). This allows for cases where the new DSN points at a different database and for cases where the same report was never saved interactively against that new ODBC DSN (provided at least one other report was saved in Visual CUT after interactively using the alternate ODBC DSN.

- Added a command line argument and an ini file option (both called **After_Export_Delay**). This allows you to delay processing after an export by the specified number of milliseconds. Typical use is for cases where Visual CUT exports to a network drive and immediately attempts to open it for further excel, Word or pdf processing. In some rare cases, a delay may be needed to allow the file write to disk operation to complete so the file becomes accessible to the next file open operation.

- Updated runtime components in the Crystal 8.5 & XI versions of Visual CUT. Users of prior versions can't use the online update option and must go through a Remove, Download & Install process in order to update their version.

Version 6.1001: Released March 10, 2009

- **An alternative email engine has been added**, which provides significant new email features. For details, see the user manual section of "***2009 Email Engine***":

    1. **You can now queue outgoing email messages in an 'Outgoing' folder monitored by a new smtpQ service**. As soon as a new file appears in that folder, the service will attempt to send it. If it succeeds, it will either move the file to a **'Sent' folder** or you can elect to have successful messages deleted. If after several minutes of repeated retries, the service fails to send the message, it will move it to an **'Undeliverable' folder**. These email messages are deposited as **.eml files that can be opened in Outlook Express and manually transferred between folders**. For example, if you copy .eml files from the 'Sent' folder to the 'Outgoing' folder, they will be emailed again (perhaps the receiver junked your original message by mistake). Among other benefits, **this allows you to archive outgoing emails and easily recover from email service disruptions**. For example, your email server may be down or your email service provider may not allow more than 100 messages per hour. For more detail, see the user manual section on "Queueing Emails & The smtpQ Service."
    2. **When sending messages to multiple recipients, if the SMTP server rejects one of the recipients as a bad email address the process still continues emailing to the other recipients**.
    3. **Supports secure connections to the SMTP server** (NTLM, CRAM-MD5, SSL, STARTTLS).
    4. **New 'Email 2' tab in the Options dialog** as well as new command line arguments (**Email_Service**:2009) allow you to:
        a) **Elect to use the new email engine** (**Email_Service**:2009).
        b) **Turn on StartTLS (SSL) functionality** in cases where the SMTP server requires it (Email_StartTLS:True).
        c) **Specify a destination for bounced messages**. (Email_Bounce_Address)
        d) **Encrypt the message** so that only the recipient can open it. (Email_Send_Encrypted:True)
        e) **Sign the message** so that the recipient can verify that you are the true sender of the message and that the message content hasn't been altered. (Email_Send_Signed:True)
        f) Support cases where the SMTP server uses integrated authentication (Email_SMTP_Domain).

- **A new PDF_Linearize command line argument allows you to web-enable pdf files so they open faster from a url link** because the 1st page is opened even before the full document has been downloaded by the browser. For detail, see "Linearize (web-enable) PDF Files."

- The install no longer attempts to create a registry entry that, in rare cases, could cause installation failures.

- When the **Rename_Printer_Jobs** options is set to True, Visual CUT now renames print jobs in the printer queue even if the printer is not the default printer.

- Added support for attaching files to email messages when embedding the HTML export in the email message body.

- Fixed an issue when dynamic print copies value is set to zero.

- Fixed an issue in the right-click menu option of 'Copy .rpt and Settings' in cases where a user attempts to copy the rpt from its mapped drive location to its UNC path.

- **Added support for exporting reports to "Report Definition" format**. Existing users may add that option by adding a record to the Export_Opt table (in Visual CUT.mdb) with "crEFTReportDefinition" as export constant and "Report Definition" as Export Name.

- Added an option (under the Process tab of the Options dialog) to turn off the transfer of files to the recycle bin before they get overwritten by report exports.

- Added a user manual section describing how you can add ***Dynamic Tables inside HTML Email Messages*****.** HTML tables are very useful, particularly when sending email messages to mobile devices. Because the width of an HTML table and its columns can be specified as percentages of the available screen width, the information adapts itself to the device displaying the message.

- Added Email_SMTP_Port, Email_User_ID, and Email_Password as a command line arguments.

- Added a button to the Export/Email tab that copies the export file text into the email attachment option. This is just a convenience instead of manually copying & pasting.

- Added code to handle cases where a user clicks on the button to 'browse for reports to open' and the last used rpt folder is not accessible to that user.

- Improved warning message for missing pdf source file in PDF_Merge operations.

- Using the **Word_Replace_Tags** command line argument, **you can now populate Word tables**. For detail, see: "Populating Word Tables with Crystal Formula Data." Also, tags are now recognized when placed in all Word document areas such as headers, footers, text frames, comments, endnotes, and footnotes.

- **By naming parameters in a certain way, you can now ask Visual CUT to automatically load their values from ini file entries**. One possible use may be for reports where some parameters change only rarely. This also **allows report developers who sell their .rpz files in vertical markets to protect their reports from unauthorized use.** It also allows **each client to customize the reports with text elements, conditional formatting, or record selection criteria without changing the report design**. For more detail, see the sections on "Load ini Values into Parameters" and "Securing Reports against Unauthorized Use." This functionality was originally developed for DataLink Viewer.

- **Added a button to the Options dialog to easily open Failure.log in Notepad**.

- **Added a button to the Options dialog (Process Tab) to make a global change in stored login information in Login_Opt table in Visual CUT.mdb.** This is useful in cases where database passwords and/or user ids must be changed.

- **Added a button to the Export/Email tab to easily open the Email Log**.

- **Improved main window resizing**. Increasing the window width now also increases the width of the list boxes showing the group values and the fields/formulas available for drag & drop. This helps in cases where fields or formula names are very long.

- **During command line invocation of Visual CUT processing you can now avoid seeing the brief minimization process of the Visual CUT window and the secondary processing window in the status bar.** This is particularly useful when calling Visual CUT from another application. To achieve this, you can edit the [Options] section of DataLink_Viewer.ini and set: **Suppress_Progress_Window_In_Command_Line_Processing** to True.

- Visual CUT XI now uses a **newer version (Service Pack 5) of the Crystal XI R2 runtime components**. Users of prior XI versions can't use the online update option and must go through a Remove & Install process in order to update their version.

Version 6.0000: Released 11/23/2008

- **Visual CUT can now merge Crystal fields/formula values into a MS Word document** containing tags referencing the Crystal fields/formulas. The resulting new file can then be saved to a new dynamically-named MS Word file (or appended into an existing target file) and emailed to dynamic destinations. This is supported during whole report processing as well as during bursting. The advantage of this technique over directly exporting/bursting a Crystal report to a Word document is that Crystal exports to Word tend to have formatting and editing problems. By creating your own Word document to act as a template, you ensure precise formatting and problem-free editing. For detail, see "Inserting Crystal Values into MS Word Documents."

- **Added Integrated Interactive Authentication functionality. This allows users to avoid repeated logins during interactive use of Visual CUT.** For detail, see "Integrated Interactive Authentication" in the user manual.

- **You can now direct Visual CUT to use a different folder for key files** (instead of the default location, which is the application folder). For detail, see the new user manual section on "File Location Functionality."

- During merging of pdf files, **Visual CUT can now generate bookmarks in the merged pdf file based on the names of the merged files**. For detail, see: 'Using the Merged File Names to Generate Bookmarks.'

- Improved PDF_MERGE handling of cases where wild card handling finds zero files to merge.

- Visual CUT XI now uses a **newer version (Service Pack 4) of the Crystal XI R2 runtime components**. Users of prior XI versions can't use the online update option and must go through a Remove & Install process in order to update their version.

- The **Print_Copies** command line arguments now also applies to scenarios where you use the command line arguments of **PDF_Print**, **PDF_Print_Split**, or **PDF_Print_Split_Tag**. The number of copies specified can be a dynamic reference to a field or a formula in the report.

- Fixed an issue (message box) related to bursting reports with comments in the record selection formula.

- Added code to handle cases where users left a space in the cc or bcc email fields.

Version 5.9001: Released 7/21/2008

- The XI version of Visual CUT is now packaged with Crystal XI R2 Service Pack 3 (SP3) runtime components. Users of prior XI versions can't use the online update option and must go through a Remove & Install process in order to update their version.

- The XI version of Visual CUT now supports the new standard (rather than legacy) csv export formatting. To use this option, select "Char Separated Values" rather than "Comma Separated Values".

- You can now use the Options dialog (**Processing** Tab, **Display Warning when email Attachment <> Export File** checkbox) to disable the warning message when the email attachment is not identical to the export file.

- Added **new Date Constants** (**Now_Plus_S** & **Now_Minus_S**), which allow specifying current datetime (for DateTime parameters) or current time (for Time parameters) shifted by a specified number of seconds. For detail, see the "Date Constants" section of the user manual.

- **A new PDF_Properties command line argument allows you to set the pdf document properties of *Author, Title, Subject, Key Words, Creator,* and *Producer***.
  For more detail, see the section "Setting PDF Document Properties via Command Line Argument".

- **A new PDF_LinkToWeb command line argument allows you to add web browser links to a web site, email address, or local file (e.g., batch file) hotspots (with or without text) to pdf files**. For detail, see "Adding Web/File/email Hotspot to a PDF File."

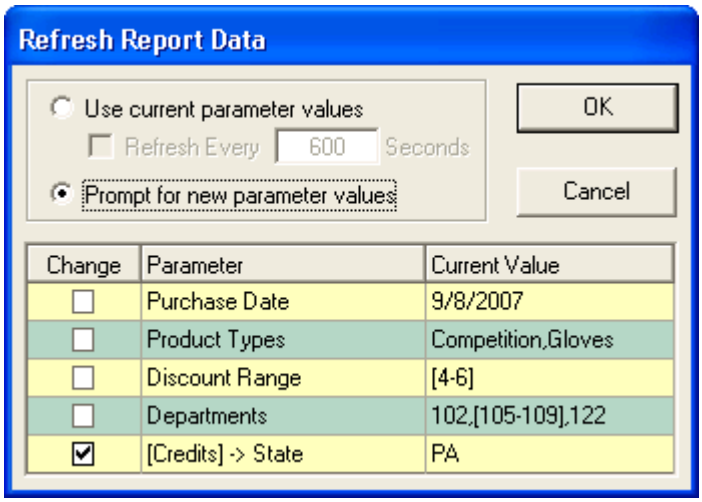
- **A new PDF_AddImage command line argument allows you to add an image to a pdf file with an optional link to a web site, email address, or local file.** This can be useful when you wish to **use a button image as a hotspot** indication or when you wish to **add a company logo to a range of pages.**

- **Visual CUT can now export reports to a multi-page TIFF image file** by automating the process of first exporting to PDF and then saving the pdf file as a TIF file. **TIF** is now a new format option (in addition to BMP, JPEG, WMF, EMF, EPS, PNG, or GIF) in the **PDF_Save_As** command line argument.
  For more detail, see "Saving PDF Files to Image Files."

- **A new PDF_MERGE_Files_to_Layers allows you to merge 1-Page PDF Files into Layers in a Single PDF File. This is particularly useful for mapping applications where you users wish to turn on or off the visibility of certain map layers**.
- Fixed an issue causing Table of Content entries in pdf files to lose their link to pages that are 90 degrees rotated and are subject to page numbering operation.

- PDF processing can now be done for very large files. With the previous version, problems could be experienced with pdf files that are larger than 150 MB.

- **Users can now select which parameters they wish to refresh**. The parameter refresh choices are stored for each report and can be easily reused or changed at a later session. Note: independent (unlinked) subreport parameters participate in this process and are listed as [subreport name] -> Parameter Name.



- Fixed a code issue causing repeated login failure messages when command line processing fails to connect due to problems such as wrong password.

- Fixed a problem causing linked dynamic parameter reports to require their own login.

- Fixed a problem in processing Time parameters.

- Fixed a problem caused by blank export file name when using Printer (Default) as the export format.

- Fixed an issue caused by users who, by mistake, separate a list of files with two delimiters (for example, ";;") instead of one delimiter (for example, ";").

- Fixed an error when using the same parameter multiple times within a dynamic parameter cascade (when using the special functionality of other reports acting as parameter dialogs).

- Fixed an issue related to command line processing of reports that have no tables.

- Fixed an issue with printing reports to the default printer when the report was designed in Crystal to use 'No Printer' (optimized for screen display).

- Fixed an issue related to bursting reports with comments in the record selection formula.

- Fixed an issue related to changing ODBC DSN data sources for subreports.

- Update.exe, the component providing online update functionality, is no longer marked as an essential component. This means that Symantec AntiVirus may (falsely) quarantine that component without causing the headaches of automatic reinstalls. In order to retain the ability to do online updates to newer versions, you can open Symantec Antivirus, expand the Configure node, select the File System Auto-Protect node, click on the Exclusions button, click the File/Folders button, and select update.exe (in the folder where you installed my software) to be excluded from the virus protection.

Version 5.8001: Released 11/25/2007

- **Added a "Copy .rpt and Settings" right-click menu option to the report grid. This "clones" the report and its settings**. The report file gets copied to a different name or folder, and all its settings (processing options, parameters, login information) are copied over. This lets you **move a report to a different folder** or **from a mapped drive to a UNC path** (\\server\folder\). This also lets you **process the same report in a different way by changing some settings for the cloned report** (as an alternative to using command line arguments to override saved processing options).

- **Suppressed the distracting export progress window that pops up on each export**. Thanks to Frank Schwarz (Orbitz) for the suggestion. Note: if you wish to see the export popup or if Visual CUT starts misbehaving due to this change, you can edit DataLink_Viewer.ini and set **Suppress_Export_Progress_Popup** to False.

- **Several enhancements to the user interface**: a) improved screen shortcuts, b) better layout to the email options tab in the Options dialog, c) the Reload button now triggers a Preview of the reloaded report, and d) a single click on the first report row in the report list grid now selects that report for preview.

- Added code to avoid rare cases of wrong status shown in the export burst checkbox.

- Fixed recognition of Email_Delay_MilliSeconds as a command line argument.

- Visual CUT now renames up to 20,000 print jobs (in the Printer Queue). The prior limit was 255 jobs.

- The option dialog now allows you to turn off both the Success.txt as well as the job status files (VC_Job_Status_Y.txt or VC_Job_Status_N.txt) functionality during unattended processing. Previously, that option controlled only Success.txt.

- Visual CUT can now open .rpz files created in the new DataLink Viewer version (5.6001 and above) as well as older .rpz files.

- You can now **specify "Default" as the printer name** in the command line arguments of "**PDF_Print**", "**PDF_Print_Split**", and "**PDF_Print_Split_Tag**".

- **You can now use up to 10 (instead of just 1) bookmark formulas for each group level**. This is particularly **useful in cases where the same group level has multiple (split) group header sections** (e.g., GH2a and GH2b) and you wish to generate a bookmark at the start of each split section. The naming requirements for the 9 extra functions are specified in the user manual under the header of: "Naming the Bookmark Formulas."

Version 5.7001: Released 9/27/2007

- Visual CUT XI now uses the Crystal XI R2 SP2 (Release 2, Service Pack 2) runtime components. To take advantage of the new runtime components, users of prior XI versions should go through a Remove (using the old msi file) & Install cycle (do not use the Online Update wizard).

- Crystal XI R2 provides two new export options: **Outlines in Excel Data Only exports** and **bookmarks in pdf exports**. You can't specify these two new export options directly in automated Visual CUT processing. Instead, open the report in Crystal XI R2 and set the options using:
  File, Export, Report Export Options…
  Note: a more powerful (control over bookmark labels and color) PDF Bookmark functionality is already provided by Visual CUT in all versions, but the new functionality allows you to avoid creating a formula for each bookmark level and is a good choice for simple cases.

- **Enhanced merging of pdf files** (using the PDF_Merge command line argument). **During the merge process, you can now add bookmarks for each input pdf file, so that the user can easily navigate the merged pdf file**. For more detail, see the new user manual section on "***Specifying Bookmarks when Merging PDF Files***."

- When using PDF_Merge, **Visual CUT now retains the original expand/collapse status, color, and style (bold/italics) of the merged bookmarks**.

- Enhanced the command line argument for generating a Table of Contents based on pdf bookmarks (PDF_TOC) so that **a bookmark is now generated and linked to the newly inserted Table of Contents**.

- Added an option to specify different text for pdf Table of Contents headers.
  For example, instead of "Table of Contents" you can use "Table des matières".
  For detail see the user manual section on ***Overriding the default "Table of Contents" Header Text***.

- Fixed image draw failures that occurred in some pdf exports.

- Improved the way page numbers are added (with the PDF_Page_N command line argument) when the pdf file contains a mixture of rotated (90 degrees) and regular pages.

- Fixed an issue when using PDF_Page_N command line argument and specifying the optional pdf file name without the optional font type.

- Improved space use in the Export/Email tab when users increase the window size.

- For export formats that have an options dialog, a new button is now visible to the right of the export format drop-down. This button allows users to easily revisit the export options dialog for the current export format (no need to change to a different export format and then select

the original one again). This applies to versions 9 & XI.

- Fixed an issue related to the Printer (Default) and Printer (Specified) export formats.

- Added a command line argument (TXT_Merge) for merging text files (before optionally emailing the resulting merged file. This is useful in cases where you wish to append text exports into existing files or when you need to add column headers to a csv export. For detail, see the section on "Merge Text Files."

- Fixed a problem on machines with Excel 2007 when bursting to multiple excel tabs.

- Fixed an Excel Tabular export problem in the Crystal 8.5 version of Visual CUT.

- After inserting worksheet tabs into excel workbooks, Visual CUT now sets the default tab to the first one in the workbook.

- Added a command line argument (XLS_AutoFilter) for turning on the **auto filter behavior in exported Excel files**.

- Added a command line argument (XLS_AutoFit) for **automatically fitting column widths to content in excel exports**.

- Added a command line (XLS_Print_Setup) to control the print setup of excel files. For detail, see the user manual section on ***Setting Up Print Properties for Excel Workbooks***.

Version 5.6001: Released 3/31/2007

- **Added support for other character sets in email messages**. In order to override the default character set (iso-8859-1) you should add an entry to the [Options] section of DataLink_Viewer.ini. For example, in order to support Chinese characters, you should add the following entry:
  `Email_Char_Set=big5`

- Enhanced email signaling for end-of-message. For one user, this solved cases where messages to Yahoo email addresses remained in the email server queue.

- **Added two new date constants: Last_MM_DD and Next_MM_DD**. For detail, see the section on "Date Constants."

- **Added a new command line argument for specifying custom email header(s)**. For detail, see the section on "Using a Command Line Argument to Specify Email Headers." The request for this feature came from a customer who needed to specify email message sensitivity (as recognized by Microsoft Outlook).

- Fixed an issue causing emailing for each group level 1 to stop if one of the groups has a missing attachment. The corrected behavior skips the emailing for the group with missing attachment and continues to process the rest the groups. Reminder: use the <opt> "marker" to designate optional attachments.

- **Added an option to instruct Visual CUT to reconnect to the email server every N messages during email bursting.** This is useful in cases where the email server limits the number of messages that can be sent within a single email connection. For detail, see the section: "Specifying an Email Reconnect Option for Email Bursting."

- **Faster export & email bursting**.

- Only for the 9 & XI versions of Visual CUT: **added export dialog options** for the Char Separated Values export format. The dialog and options (string delimiter, field delimiter, number formatting, date formatting) behave just as within Crystal 9 and XI. To support this change, existing users should add one more column to the **Report_Export_Options** table in the **Visual CUT.mdb** Access database:
  Field Name: CharStringDelimiter
  Data Type: Text
  Description: Sets the character used to enclose Strings. The default is double quotes.
  Field Size: 1
  Default Value: """" (4 double quotes specify a default value of 1 double quote)

- Fixed a problem when opening .rpz files (.rpt files encrypted into .rpz files).

- Fixed a problem related to not recognizing zero selected records due to a Group Selection formula.

- Increased field sizes of UserID and Password in Visual CUT.mdb from 50 to 250 characters. This is needed since the new encryption encoding introduced in version 5.5001 generates longer strings. Existing users should open Visual CUT.mdb (using MS Access), open the Login_Opt table in design mode, and increase the field size for UserID and Password from 50 to 250.

- Fixed an issue (limited to the XI version of Visual CUT) related to scheduling printing of reports that were designed in Crystal to use a specified, rather than default, printer.

- **Improved pdf processing speed** (bookmarks, table of contents, password protection, page numbering, pdf file merging, etc.), **reduced size of exported pdf files** (compared to pdf exports directly from Crystal), and **enabled processing of extremely large pdf files**.

- **Added wildcard functionality for specifying list of pdf files to merge**.

- PDF_Print_Split_Tag functionality is now available in all versions of Visual CUT.

- Added a warning message when a user attempts to specify an email 'Reply To' option that includes a regular name. Email processing may fail if the 'Reply To' option contains more than just an email address.

- Added a warning message when a user saves non-bursting settings for a report, but uses dynamic fields and formulas (those you drag & drop into various options in Visual CUT) from **Group Header/Footer** sections in the report. When processing whole reports in a single step via a command line, dynamic fields/formula values are recognizes only from **Report Header/Footer sections**. This is designed to avoid wasting time during processing of scheduled reports. If you are not bursting, then you are processing the WHOLE report in one step -- hence, the dynamic field information should come from report-level sections rather than from Group-level sections.

- When processing an **<<Insert_File:…>>** token, after the content of the file is inserted into the email message body, Visual CUT now searches for and replaces references to fields and formulas with their dynamic values from the report.

- Fixed database login issues with reports that use multiple data sources, one of them being MS Access direct (DAO) connection.

- Failure to connect/logon to a database during command line or scheduled processing (with Silent Failure) is now logged and aborted instead of causing a login dialog.

- Added an optional Font Type parameter to the PDF_Page_N command line argument.

- When triggering printing for report with no saved settings, the number of copies now defaults to 1. This avoids the need to specify "Print_Copies:1" as a command line argument.

- Fixed an issue with bursting reports that have more than 32,768 Groups at level 1.

- **Added a button (to the right of the Scheduling String) to automate the process of inserting the command line into a new or existing batch file**. The process also allows the user to **automatically open and inspect the resulting batch file in NotePad**. The folder used for the last batch file creation/update is stored as a default for future batch file operations.

- **Added a way to specify multiple printer destinations via a text file**. For detail, see the section on "Using a Text File to Specify Multiple Printers."

- **Added a command line argument (PDF_From_TIFF) for importing multi-page TIFF files into pdf files**. A typical use scenario is to combine scanned images (e.g., shipping documents) into PDF file exports (e.g., invoices) before emailing to customers.

- PDF Exports can now generate Form Fields based on formulas placed on the report. This means that you can use Crystal Reports to design pdf forms, and Visual CUT to generate and distribute these forms. This functionality is triggered by a new command line argument (PDF_Form_Tags) for adding form fields to PDF exports. For more detail, see the user manual section on "Adding Form Fields to PDF Files." You can download a sample of such a pdf export with form fields at:
  www.milletsoftware.com/Download/Visual_CUT_PDF_Export_with_Form_Fields.pdf

Version 5.5001: Released 10/20/2006

- **You can now invoke processing for reports with more than 8 parameters by providing the parameter values via command line arguments**. You could always override up to eight parameter values saved for the report within Visual CUT by specifying them in the command line. This was limited to the first 8 parameters in the report. This release allows you to **specify via a command line the values for an unlimited number of parameters**. For example, if you need to invoke processing for a report with 11 parameters, your command line may now end with:
  ...."Parm9:Value" "Parm10:Value" "Parm11:Value"

- **Changed encryption encoding to ensure reading and writing encrypted values is not vulnerable to special characters** (e.g., new line character). The change occurs on the fly when you close the Options dialog or log in to your database.

- Fixed a problem causing the list of reports in the 1st tab grid to be deleted when early command line processing encounters an error.

- Added a command line argument and an ini file option (both called **Email_Delay_MilliSeconds**). This allows you to delay email processes by the specified number of milliseconds. Typical use is for cases where Visual CUT exports to a network drive and immediately emails the file as an attachment. The network drive may require some time to recognize the new file.

- PDF merging now occurs before rather than after the generation of Table of Contents.

- Added date constant of **Nth_N_PLUS_M** and **Nth_N_MINUS_M**.
  For example, Nth_16_MINUS_1 returns the 16th of the prior month.
  See "Date Constants" for more detail.

- Corrected <<**File_Insert:**…>> to <<**Insert_File:**…>> in the user manual.

- Added support for printing as an export destination. This allows print bursting to be initiated/stopped interactively. If the Rename_Printer_Jobs option in the DataLink_Viewer.ini file is set to True, the printer queue shows the group value for each burst cycle print job. Existing users should add two new rows to the **Export_Opt** table in the **Visual CUT.mdb** Access database:

Export Constant	Export Name
Printer_Default	Printer (Default)
Printer_Specified	Printer (Specified)

- Added a warning message with option to abort in cases where users specify an export file that is identical to, and hence would overwrite, the report file being processed.

- Fixed printing issues in the Crystal XI version.

- Fixed After_Burst_Batch processing in cases where Skip_Recent is used and the report is not sorted by time.

- **Enhanced Email alerts about processing failures**. Such alerts now include cases where a failure occurs early in the command line processing stage. The email text now includes information about the report being processed.

- **Added a Default SMTP Server to the Options dialog**. This removes the need to manually enter an SMTP server for each report (in cases where Visual CUT fails to automaticallydetect the correct SMTP server in your network environment). This also allows email alerts about processing failures at early stages of processing.

- You can now use a dynamic SQL query to populate the Email to, cc, or bcc with a list of email addresses. See "Specifying Email Distribution Lists in SQL Queries" for more detail.

Version 5.4001: Released 7/30/2006

- **Fixed an issue causing scheduled processing failure when no user is logged in**. Users no longer need to use a more advanced scheduler such as AutoTask 2000 instead of the Windows task scheduler to address such scenarios.

- You can now **insert file content into the email message body** using a text token (or Crystal formula) that references the location and name of that file. For detail, see "Embedding File(s) Content in Email Message Body".

- Fixed logging to ODBC issues related to: a) having email addresses with embedded single quotes, b) using Skip_Recent command line argument, c) updating "Started" status to "OK" when emailing.

- Fixed a failure notification loop when the email notifying about the error fails itself.

- Fixed an issue when exporting to multiple formats in a single pass (export file name is specified as multiple files separated by ";") and one of the formats is html.

- Only for Visual CUT XI: added Excel export option to control whether Grid Lines show in excel exports. Existing Visual CUT XI users should add a new column to the **Report_Export_Opt** table in Visual CUT.mdb. The column name is **ExcelShowGridLines**, Data type: Yes/No, Default value: No.

- Added two variations to the Date Constant **Today_Minus_N_Minus_M**:
  Today_Minus_N_Minus_M_SOM returns the Start of Month for the resulting date.
  Today_Minus_N_Minus_M_EOM returns the End of Month for the resulting date.

- You can now use the Options dialog (**Processing** Tab, **Display Bursting Requirements (if not met)** checkbox to disable the "In Order to Enable Electronic Bursting your report must have…" warning message.

- The **progress window now show the % Complete in its title**. This allows users to monitor progress even when the window is minimized.

- Command line arguments can now have embedded double quotes. For example: "Email_To: "Ido Millet" <Ido@MilletSoftware.com"

- When using the **TabInOldFile** option in Excel exports, if the tab already exist the **export gets appended to the end of data in the existing tab**. In the past, such a situation would result in adding a new TabName(2) tab.

Version 5.3001: Released 6/3/2006

- Added an option to **send an email alert to a specified address when a failure occurs**. See the new Log/Alert tab in the Options dialog.

- Added options to **log processing (including bursting detail) to a table in a specified ODBC DSN**. See the new Log/Alert tab in the Options dialog. For detail see "Record Processing to an ODBC Database".

- You can now control how many levels of PDF Bookmarks are visible when a user initially opens the exported file in Adobe Acrobat. For detail, see "Controlling How Many Bookmark Levels Are Initially Expanded".

- Fixed an issue causing the completed Visual CUT process to remain open when the scheduled process has no access to a desktop.

- Fixed an issue with PDF bookmarks.

- Visual CUT XI only: Added a "**Microsoft Word – Editable (RTF)**" export format option. Existing users can simply use the "Rich Text" Format (which now exports to the editable RTF format) or add the following record to the **Export_Opt** table:
  Export Constant: **crEFTEditableRichText**
  Export Name: **Microsoft Word - Editable (RTF)**

- Fixed a bursting issue when record selection formula contains comments as last lines.

- Fixed an issue with Excel Data Only exports in the Visual CUT 11 version.

- Fixed an issue with exporting to multiple file formats in a single pass (by specifying multiple export file names separated by a semi-colon).

- Fixed a processing issue with the Printer_Burst_Only command line argument.

- Fixed a recently introduced issue with multi-value parameters.

- Enhanced the user manual and the sample reports to better explain how exported pdf bookmarks can be color coded using Crystal formula logic and how bookmark formulas should guard against Null values.

- Fixed embedding of HTML 3.2 exports with multiple images in email messages.

- The email password specified in the Options dialog is now stored encrypted in the DataLink_Viewer.ini file.

Version 5.2001: Released 3/5/2006

- Added a command line argument (ZIP_Files) to **ZIP and password protect any number of files**. This can be particularly useful when emailing files that need to be protected or combined into a single attachment.
  For detail, see "ZIP and Password Protect Files."

- Added a command line argument (XLS_Protect) to **password protect Excel files**. For detail, see "Password Protecting Excel Workbooks."

- Since including the current date in export file names is a common need, Visual CUT now adds to the Fields & Formulas area a **list of current date values (day, month, year) with various variations (number, name, length).** This allows you to avoid creating formulas in your report just to pass current date values to Visual CUT.

- **When specifying multiple file attachments for emailing, you can now specify the full path of only the first file**, if all other files are at the same folder location.

- **You can now specify that an email attachment is optional** by prefixing **<Opt>** to the file path & name. For example, if you need to email Sales.pdf (always) and Returns.pdf (if it exists), use: **c:\temp\Sales.pdf;<Opt>Returns.pdf**

- You can now **control email message priorities** using a command line argument (Email_Priority). For detail, see "Using a Command Line Argument to Specify Email Priority."

- Added a **new parameter date constant** (Today_Minus_N_Minus_M) **for subtracting days and months from the current date**. For detail, see "Date Constants" on page 65.

- **Added an "ODBC_DSN_From_To" command line argument**.
  This is useful for cases where a report uses multiple ODBC DSNs and you need to override only one of them. For detail, see: "Database Choice Functionality (Command Line / GUI)" on page 76.

- Fixed a multi-tab excel export issue (problem was limited to the Crystal 9 version).

- The PDF_MERGE command line argument now allows you to specify the full path of only the first file in the list of files to be merged, if all other files are at the same folder location.

- Added an option for using a text file to provide the list of file names for PDF_MERGE operations. For detail, see "Using a Text File to Specify Files for Merging" on page 132.

- PDF-related command line arguments are now processed even if the report is not exported to pdf and even if the "PDF Exports include Document Properties or Bookmark Processing" option is disabled.

- Fixed a problem in processing reports that have a single quote in their rpt file name.

- Fixed a problem in saving settings for reports with inactive parameters.

- Various improvements to the user manual, including a new **table summarizing the functionality of all available command line arguments**.

- Added an option to avoid attempts to connect to the database without a password.
  By default, Visual CUT first attempt to connect to the database without a password. If all your tables require a password, change the following option in the DataLink_Viewer.ini file from the default of TRUE to FALSE.
  This would reduce potential lockouts and failed login entries in the DBMS log.
  [Options]
  Attempt_Logon_Without_Password=FALSE

  Note: if some of your data sources do not require authentication, turning this option to False would trigger a logon dialog in Visual CUT when you run a report against them. Simply click OK in the login dialog without providing a user id or password.

- Fixed a problem with handling record selection formulas with embedded comments.

- Fixed a problem (introduced in prior version) with decimal parameter values.

- Fixed a problem with the starting page parameter controlling the generation of Table of Contents in PDF files (PDF_TOC command line argument).

- Visual CUT now generates a failure log message when the printer specified in a command line argument is not found. The message lists all the printer names that Visual CUT found as well as the attempted printer name.

- During Printer_Burst_Only processing, Visual CUT no longer removes target export file names to the recycle bin (since the report is only printed, there is no danger that exports would overwrite important files).

- During Printer_Burst_Only processing, **Visual CUT now names each print job according to the export file name for each group**. Previously, the print job name was fixed as the export file name belonging to the 1st group.

- You can now **specify "Default" as the printer name** in the command line arguments of **Printer**, **Printer_Only**, **Printer_Burst**, and **Printer_Burst_Only**.

- When selecting **TEXT export format**, a new dialog allows you to **control the Characters Per Inch**. By increasing this value you can ensure that exported text doesn't get truncated on the right.

- Added a command line argument (PDF_Save_As) to convert PDF files into image files (BMP, JPEG, WMF, EMF, EPS, PNG, or GIF). For detail, see "Saving PDF Files to Image Files."

- Significantly faster processing of: a) **reports that generate many pdf bookmarks**. b) **pdf exports** (when the "PDF Exports Include Document Properties and Bookmark Processing" is turned on**)**, and c) **pdf merge operations**.

- Added a **PDF_Print_Split_Tag** command line argument. The advantage of this option over the older **PDF_Print_Split** argument is its ability to dynamically control paper trays using formula logic from within Crystal reports. For example, it makes it easy to print the first and last pages in each group using one tray and print the rest of the pages from another tray. For detail, see "Using PDF_Print_Split_Tag" on page 140.

Version 5.1001: Released 10/14/2005

- **Faster loading and bursting of large reports with many group values**.

- Visual CUT can now **embed TEXT exports (or any specified .TXT, .TEXT, or .PRN attachment file) directly in the email message body** (instead of as an attachment). See "Embedding TEXT Export in Email Message Body".

- After a PDF Merge, if there are bookmarks anywhere in the resulting pdf file (even if the first document doesn't have bookmarks), the bookmark window is made visible.

- Fixed a problem causing some command line arguments in the Crystal XI version of Visual CUT to be treated as static rather than dynamic.

- **Warning messages are now not logged** into the Failure.log file. A new checkbox ("Log Warnings") in the Options dialog allows you to request logging of warnings.

- **Email attachments can now be specified using wild cards** in the file name(s).
  For example, to send all files that start with the current Customer ID:
  C:\VC_Exports\{customer.customer_id}*.*

- Data Constants for date and numeric parameters are now supported for range (not only discrete) parameters. See "Using Command Line Arguments to Specify Parameter Values".

- When using **Printer_Burst** or **Printer_Burst_Only** command line arguments, the **printer name argument is now dynamic.** You can create a formula that returns a different printer name for different group values, embed that formula as the printer name argument, and sent the printouts for different groups to different printers.

- For ODBC data sources, users can now directly **enter a database name into the login dialog** if they wish to override the database specified in the ODBC data source or in the report itself. For detail see "Overriding the Database Specified in the Report or ODBC DSN."

- When using a Native connection to Oracle, you can now change the Server name in the login dialog. Also, a new command line argument (Oracle_Server) allows you launch report processing against a specified Oracle server. See "Overriding the Server in Native Oracle Connection."

- Visual CUT can now merge formula values into a PDF Form and save/send the result as new PDF files. See :Saving PDF Files to Image Files."

Version 4.9007: Released 7/15/2005

- **PDF Merge operations now also merge the bookmark information from all files**.
  The details are discussed in the section "Merging pdf Files"
  In particular, read the section on "Using a Text File to Specify Files for Merging."

- Fixed an ini file problem causing the Set_PDF_Properties option to be set to TRUE each time Visual CUT was reloaded.

- **Fixed a multi-tab excel export issue** (when using the Tab! Option).

- Fixed an issue causing Export_Mode and Email_Mode command line arguments to be ignored if the report has saved settings.

- Fixed an issue causing reports with no saved settings to abort processing if no export or email options were specified (for example, in cases where only a printout is desired).

- **Removed internal validation of email addresses** (in case users are using an email-to-fax bridge which requires a non-standard email address).

- Fixed an issue causing generation of PDF bookmarks to ignore cases where the report is exported in whole (no bursting) and there's **only one requested bookmark**.

- Added an option to **start page numbering of PDF files on a given page and a given number.** For example, start numbering on page 2 with the number 1. See the section on Adding Page Numbers to a PDF File for more detail.

- Added an **option to adjust page numbers in generated TOC**. For example, if page numbering for the document started with 1 for page 2 (ignoring a title page), then the TOC can also offset the displayed page numbers by 1. See the section on "Adding a Table of Content to the exported PDF File" for more detail.

Version 4.9003: Released 4/25/2005

- The ODBC Export Options dialog now allows you to select **Abort** (default Crystal action), **Append**, or **Replace** as the action when the target table already exists.
  The details are discussed in the section "Text Functionality."

- Using command line arguments, **you can now invoke processing even for reports that have no saved settings in Visual CUT**. The details are discussed in the section "Using Command Line Arguments to Process Reports with No Settings."

- New **Export_Format** command line argument is now available to provide or override the Export Format saved for the report in the 3rd tab options.

- New **Export_Mode**, **Email_Mode**, and **Email_SMTP_Server** command line arguments are now available to provide or override the processing options saved for the report in the 3rd tab options.

- The **Export_File** and **Email_Attach** command line arguments are now **dynamic** (field/formula names are replaced with values from the report).

- **You can now export to multiple files and export formats in a single pass**.
  This applies even to bursting, so each bursting cycle can generate multiple export files and formats. The details are discussed in the section: "Exporting to Multiple Files/Formats in a Single Pass."

- Fixed a login problem (when subreports connect to a different data source). This problem occurred only in the Crystal 8.5 runtime components version of Visual CUT.

- Fixed a problem in changing ODBC DSN sources when running the Crystal 9 runtime components version of Visual CUT with reports from earlier versions of Crystal.

- Added a **TabInOldFile_Replace!** Option in excel export file names.
  Visual CUT inserts the exported report as a sheet (tab) in an existing excel file if the tab doesn't yet exist, but **replaces the content of the tab if it already exists**.
  This allows other sheets to refer to the content (periodically refreshed by Visual CUT) of that tab. The details are discussed in the section: Appending or Replacing Data for Existing Tabs**.**
  Thanks to Egil Stenberg (Flextronics) for sponsoring this development.

- The installer(msi) file now **defaults to a per-machine installation, if installed by an Administrator**. This facilitates the deployment of Visual CUT when an Administrator installs the software for use by other users. In the past, such cases would require installation from a command line with an ALLUSERS=1 argument.

- Visual CUT 9 now uses a new set of Crystal 9 runtime components that, among other minor fixes, can provide **much faster execution of Crystal 8.5 reports** in certain cases (one user reported a decrease **from 3-5 minutes to 10 seconds**).

- Added a command line argument (PDF_Print_SPLIT) that allows you to **split the printout of a PDF file to multiple printer trays**. For example, you may want to print the first page from a tray containing your company's letterhead, and the rest from a regular paper tray.

Version 4.6005: Released 2/17/2005

- **New Grid** for Selecting Reports with enhancements such as:
  - **more columns** (path, report, title/description, **subject**, **type**, date/time **last used**)
  - **sort on any column by clicking the column header**
  - **group on any column by dragging the column header to a Grouping area**
  - **select larger or smaller font size** (right-click column headers…)
  - **hide/show columns** (right-click column headers…)
  - Select multiple reports and load them into the grid in one step

- When running reports that use ODBC data sources, you can **select which ODBC data source should be used**. This is explained in detail in the section: Selecting an Alternative ODBC Data Source .

- **A new command line argument (ODBC_DSN) allows external control of which ODBC data source is used**. This is explained in detail in the section: Selecting an Alternative ODBC Data Source .

- The **main window remembers its size from session to session**.

- **Added a browse button to allow easy selection of file attachments.**
  The dialog allows selection of multiple files and adds the choices (separated by semi-colon) to any choices already specified. Note: if file attachments are dynamically named (using fields/formulas from the report), you must edit the resulting text because the file browse dialog is restricted to existing files.

- **Added a browse button to allow easy selection of a text (*.txt) file for use as an email distribution list.** After the file is selected, Visual CUT automatically adds the required "FILE:" prefix inside the "Email To" option.

- Added **Number constants** for specifying Year or Month relative to the current year or month. This is described in the section "Number Constants." Thanks to Paul MacKenzie (Silberline) for sponsoring this development.

- Added a **Skip_Recent** command line argument allowing you to instruct Visual CUT to skip processing of a whole report or a bursting step if the target export file already exists and was created less than N minutes ago. A detailed description of how this allows you to avoid generating duplicate emails is described in the section "Avoiding Duplicate Processing."

- Visual CUT can now **embed HTML exports directly in the email message body** (instead of as an attachment). This allows you to easily generate professional looking email messages that contain logos and charts (invoices, management reports, etc.).

Version 4.1001: Released 12/30/2004

- To facilitate emailing to a long list of recipients, you can now **specify email distribution lists** in text files. The details are discussed in the section: ***Specifying Email Distribution Lists in Text Files* .**

- Visual CUT can now handle cases where **multiple composite email destinations** (display name as well as email address) are specified in the To, CC, or BCC emailing options. The details are discussed in the "Dynamic Tables inside HTML Email Messages" section.

- Visual CUT now creates the directory of a specified merged PDF target file if the directory doesn't already exist.

- PDF Merging operations now succeed even when the pdf file names contain commas.

- Fixed a problem with Print_Copies command line argument.

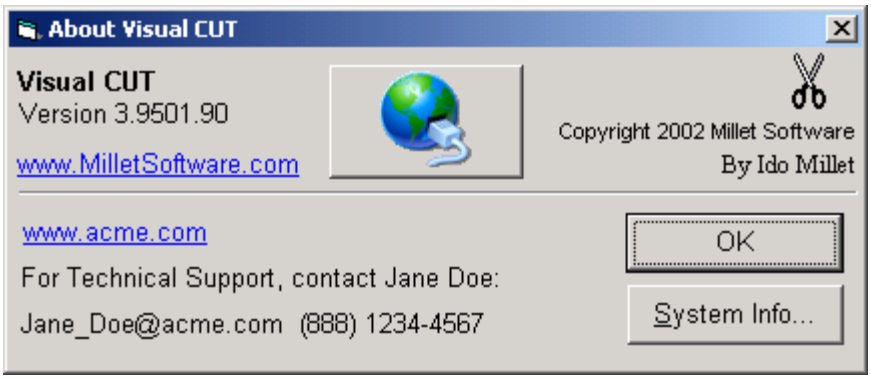
- Added DataLink_Viewer.ini options to **set 3 lines in the About dialog to text of your choice. This is useful if you are a consultant installing Visual CUT for a customer and you wish to specify your contact information for technical support.** For example, using the following settings in DataLink_Viewer.ini:

  _____

  [Options]
  About_Line1=www.acme.com
  About_Line2=For Technical Support, contact Jane Doe:
  About_Line3=Jane_Doe@acme.com (888) 1234-4567

  _____

  you would get the following About dialog:



Note: as demonstrated with the **www.acme.com** line, entries containing only a url are automatically treated as web links.

- Fixed an issue related to not storing/reusing login information (encrypted) for reports if they use the same data source and were opened one after the other in the same interactive Visual CUT session. Thanks to Paul Stewart (Worthington Aviation) for identifying the issue and testing the new version.

- Using command line arguments, you can now **override the server and the database the report connects to**. This is currently limited to SQL Server OLE DB data connections.

Version 3.9005: Released 09/29/2004

- Visual CUT can now **merge (and print if desired) any number of PDF files**.
  This allows users to combine (for e-mailing, print stapling, etc.) Crystal outputs from multiple reports with different page orientations.

- Visual CUT now treats the following command line arguments as "dynamic" in the sense that field/formula names are replaced with dynamic values from the report (just as you drag & drop fields & formulas into the various options in the 3rd tab within Visual CUT):
  PDF_MERGE
  PDF_PRINT
  PDF_PROTECT
  PDF_TOC
  PDF_PAGE_N

  Among other things, this allows you to **easily protect individual PDF exports with different passwords for each group.**

- You can now **specify the PDF file name** that should be processed by the command line arguments of: **PDF_PROTECT**, **PDF_PAGE_N**, and **PDF_TOC**.
  This allows you to add a Table of Content, add Page Numbers, and protect any PDF files -- **not just the PDF file being exported by Visual CUT**.
  The default remains to apply these processing steps to the exported PDF file, but if you specify a file name (as the last element in these command line arguments), the requested PDF processing would be directed to that file.

For example, this is useful when using Visual CUT to merge PDF files (using **PDF_MERGE**) and then protecting the resulting merged file.
Note: this is an optional argument – no need to change existing scheduling strings.

Version 3.8001: Released 08/18/2004

- Visual CUT can now generate **Color-Coded Bookmarks in exported PDF files**.
  See example at: www.milletsoftware.com/PDF_with_Color_Coded_BookMarks.jpg
  The details are discussed in the section: Controlling PDF Bookmark Colors (& Text)**.** This is useful for visually indicating item performance or status (e.g., red color indicating poor performance).

- A new command line argument allows you to **Encrypt & Protect Exported PDF files** (using 128-bit encryption). This allows you to control various aspects such as whether the user would be prompted for a Password and what actions (Print, Copy, Edit, add Annotations,…) are allowed.

- A new command line argument allows you to **Add a Table of Contents (including hyperlinks to the appropriate pages) to the exported PDF file**.
  See example at: www.milletsoftware.com/PDF_with_TOC_and_BookMarks.jpg
  This is useful when users need to navigate through hardcopy printouts of large pdf files.
  Thanks to Jim Stickley (TraceSecurity) who sponsored the development of this functionality.

- A new command line argument allows you to **Add Page Numbers to the Exported PDF file**.
  This is useful when adding a Table of Contents (as discussed above) since in such cases the page numbers from Crystal would be wrong. The details are discussed in the section: Adding Page Numbers to a PDF File .
  Thanks to Jim Stickley (TraceSecurity) who sponsored the development of this functionality.

- A new DataLink_Viewer.ini file option (**Job_Status_Path**) under a **[File_Locations]** section allows you to override the default location (Visual CUT application folder) for the Job Status indicator files.

- A new DataLink_Viewer.ini file option (**Rename_Printer_Jobs**) under a **[Options]** section allows you to override the default behavior whereby Visual CUT assigns the (dynamic) export file name option (if specified) to the document name shown in the Printer queue when executing scheduled or command line printouts of reports:

  [Options]
  Rename_Printer_Jobs=FALSE

  This is useful in situations where you don't have permissions to name printer jobs.

- Fixed an issue with paper source (tray) not being preserved when specific printer destination is specified via command line arguments.

- Fixed failed attempts to save settings when changing the number of print copies.

- **New command line arguments** are now available to override the values specified for several 3rd tab options:

  Export_File
  Email_To
  Email_From
  Email_Reply_To
  Email_CC
  Email_BCC
  Email_Attach
  Email_Subject
  Email_Message

- Visual CUT can now open and process **rpz** files (rpt files encrypted by DataLink Viewer). This allows developers to protect and hide their reports designs (either as an intellectual property issue or as a tech support issue). They can simply keep the rpt files and distribute only the **rpz** files (created in DataLink Viewer) to their users.

  Within DataLink Viewer as well as Visual CUT, **rpz** files behave just like rpt files except that, in order to protect the report design, exporting **rpz** files to "rpt" format or to "report definition" format is blocked.

Version 3.4301: Released 05/15/2004

- Fixed an issue causing processing of Printer_Burst or Printer_Burst_Only arguments to trigger an initial full printout before executing the requested Printer Burst printouts.

Version 3.4201: Released 05/06/2004

- Added functionality that allows adding Excel Exports (and/or bursts) as Tabs in existing spreadsheets. This allows you to create and email multi-tab excel workbooks that contain information from multiple Crystal reports. For detail see the new user manual section on "***Adding Excel Exports as Tabs in Existing Spreadsheets (Briefing Books).***" Thanks to Richard Roper (AIG) for sponsoring this development.

Version 3.4001: Released 04/26/2004

- Fixed a problem with "Printer:" as a command line argument.

Version 3.3901: Released 04/25/2004

- Fixed a problem that caused an error message and update failure in some situations after clicking the Export check box and then the SAVE button.

Version 3.3801: Released 04/18/2004

- Visual CUT now prompts for, and uses, **3 more options for Excel (Data Only) exports: *Export Object Formatting, Maintain Column Alignment,* and *Export Images*. Note:** Applies only to the CR 9 runtime components version of Visual CUT.
  Thanks to Larry Bodie (Kelsan) for suggesting this enhancement.

Version 3.3701: Released 04/06/2004

- **DateTime parameters can now accept the same date constants that previously were available only for Date parameters** (for example, TODAY_MINUS_**N,** START_MONTH_PLUS_**N,** END_MONTH_MINUS_**N,** START_YEAR_MINUS_**N**).

  When using these constants for a DateTime parameter, the time value is set to the start of that date (12:00:00 AM). As a reminder, these date constants are described in the "Date Constants" section of the user manual.

Version 3.3601: Released 03/27/2004

- During unattended/scheduled processing, Visual CUT now generates a **job status text file**, located in the Visual CUT application folder and named:
  VC_Job_Status_N.txt (containing the error message) if a failure occurred and
  VC_Job_Status_Y.txt if processing was successful.

- Replaced the Online Update component (Update.EXE) with a newer version (fixing an issue with NT 4.0 PCs).

Version 3.3501: Released 03/24/2004

- Fixed an issue whereby changes to the export/e-mail settings in the 3rd tab where saved even without clicking the SAVE button (if the user then selected another report). Thanks to Jim Woodin (*Diamond V Mills)* for identifying the issue.

Version 3.3301: Released 03/17/2004

- Visual CUT now assigns the (dynamic) export file name option (if specified) to the document name shown in the Printer queue when executing scheduled or command line printouts of reports. Note: to enable this functionality, the "Manage Documents" security option for the printer must be turned on for you.

- Added more informative error messages when bursting fails due to the record selection formula containing only a comment.

- **Added an option (in the options dialog) for specifying what Visual CUT should do in scheduled/unattended execution of reports that were Saved with Data.** By default, the option is set to use the saved data. By turning this option (check box) off, you can now indicate that Visual CUT should refresh the report with data from the database.
  Thanks to Ron Ruth (TIB Bank of the Keys) for suggesting and testing this enhancement.

Version 3.3001: Released 03/08/2004

- Added a variety of **new date constants** (for example, TODAY_MINUS_**N,** START_MONTH_PLUS_**N,** END_MONTH_MINUS_**N,** START_YEAR_MINUS_**N**) for specifying date parameters values relative to the date the scheduled report actually runs.

  These date constants are described in a new section ("Date Constants") in the user manual.

  Thanks to David Leland (Johnson Corporation) for suggesting this enhancement.

Version 3.2001: Released 03/04/2004

- Fixed an issue causing some changes in the Options dialog to not be saved (to the DataLink_Viewer.ini file).

- Replaced the Online Update component (Update.EXE) with a newer version.
  Note: this requires that you select the option to restart the PC when you apply this update via the online "Check for Updates" process (since Update.EXE would be detected as a "locked" component during that process.

- **Updated the PDF properties/bookmarks processing dll.** My testing shows that the 8.5 runtime components version of Visual CUT can now generate PDF bookmarks when exporting reports with images/charts without getting a warning about font substitution when opening the PDF file.
  To avoid confusion, note that this problem has never affected the Crystal 9 runtime components version of Visual CUT.

- Visual CUT can now **handle cases where the correct User ID or Password for a secure data source are actually blank.** Thanks to Tom Cook (Grotenhuis) for identifying the issue and testing the fix.

Version 3.1001: Released 02/19/2004

- Fixed an issue causing an error message when **setting excel export options for a report with no groups.** Note: this was an issue only with the CR 9 runtime component version.

- Fixed a problem when **bursting to multiple tabs within a single excel file.**
  Note: this was an issue only with the CR 8.5 runtime component version.

Version 3.0000: Released 02/10/2004

- **New User Interface** in the Select Report (1st) Tab:
    1. A grid shows three columns: **Report Path**, **Report File Name**, and **Report Title** (from the summary information specified for the report during Design Time) or a user-specified **Description** (entered by the user by clicking in that cell).
    2. Right-Clicking a row on the grid provides a **popup menu** with '**Preview Report**' and '**Delete Row'** options.
    3. The **grid columns resize** when the viewer gets resized by the user.
    4. The **grid columns can also be resized by the user and "remember" their widths** from session to session.
    5. The data is still maintained in ReportList.txt and **the new version recognizes and migrates the data in an old ReportList.txt files automatically**.
    6. **Selecting a report to open via the "Browse for a Report to Open' button causes an immediate preview** (no need to select the Preview Tab).
    7. **Double-Clicking a report in the grid of previously opened reports causes an immediate preview** (no need to select the Preview Tab).

- Added a button in the Version dialog allowing users to **check for software updates on my web site and apply those patches quickly and efficiently**. The patch sizes are typically very small since this automated process applies only net changes to files.

- **Added a button to the 1st tab for launching an Options dialog (instead of manually editing DataLink Viewer.ini).**

- The 8.5 runtime components version of Visual CUT can now generate PDF bookmarks when exporting reports with images/charts without causing a PDF file corruption. You may still get a warning about font substitution when opening the PDF file.
  To avoid confusion, note again that this problem has never affected the Crystal 9 runtime components version of Visual CUT.
  Older 8.5 versions can always avoid this issue by setting the "Set_PDF_Properties" ini file option to FALSE (as described on the last page of the user manual).

Version 2.8.7000: Released 01/24/2004

- A special email processing case that in the past would generate a message of:
  "***the attempt to connect timed out***" is now handled internally to avoid e-mail failures.

- Added support for **overriding the number of attempts to connect to the email server** (default is 4 times). If your wish to override the default, open **DataLink Viewer.ini** (located where you installed ***Visual CUT***) and set a new value (**between 1 and 20**) to the following entry under the **[Options]** section:
  Email_Connect_Retries = 4

- Added support for **overriding the number of seconds Visual CUT waits for email connection before giving up** (default is 40 seconds). If your wish to override the default, open **DataLink Viewer.ini** (located where you installed ***Visual CUT***) and set a new value (**between 5 and 120**) to the following entry under the **[Options]** section:

Email_Connect_Timeout = 40

- Added support for **overriding the number of seconds Visual CUT waits for an email messages to complete before giving up** (default is 60 seconds). If your wish to override the default, open **DataLink Viewer.ini** (located where you installed ***Visual CUT***) and set a new value (**between 10 and 1800**) to the following entry under the **[Options]** section:
  Email_Message_Timeout = 60

Version 2.8.6000: Released 01/19/2004

- Added functionality for adding bookmark information from a text file (**PDF_Bookmarks,txt**) to exported PDF files. Using my CUT Light UFL, this can be used to generate bookmark information from within subreports.
  For detail see the section

Version 2.8.5000: Released 12/19/2003

- The "**To:**" and "**Copy To:**" email options can now be structured as a **name enclosed in double quotes**, followed by the **e-mail address enclosed in "< >"**
  For example: **"Ido Millet" <ixm7@psu.edu>**

- Added special handling for Stored Procedure parameters with Null values.
  Null parameter values are now stored as "[VC_NULL]" in the report option table.
  Passing a Null value via a command line argument should be done as "Parm1:[VC_NULL]"
  Thanks to Sharon Mone (Fujitsu) and Greg Scharer (Upstate) for identifying the issue and testing the fix.

- Fixed "dynamic copies" option not being saved. Thanks to Jonathan Washam (Independent Portfolio Consultants) for identifying the issue and testing the fix.

- Changed Excel exports in the Crystal 8.5 runtime version of Visual CUT to use the "Column Headings" option by default (to match the default behavior in Crystal 8.5).
  Thanks to Greg Scharer (Upstate) for the suggestion.

- Fixed **Excel "Data Only"** exports behaving (on some PCs) as regular **Excel** exports. Thanks to Chris Kell (Washington State University) for identifying the issue.

- **Excel "Data Only" exports** now set "Column width based on the objects in the" **Report Header** (the default used by Crystal 9). Applies only to the Crystal 9 runtime component version of Visual CUT.

- You can now **specify export options** for ***Excel, Excel (Data Only)***, and ***Paginated Text*** exports using dialogs like those you get when exporting from Crystal 9.
  Applies only to the CR 9 runtime components version of Visual CUT.
  Thanks to Bernard Paes (Flag Choice Hotels) and to Ken Rickard (EMU) for suggesting this enhancement.

- **Added the ability to export to ODBC.** A new dialog collects options such ODBC DSN, user id, and password (stored in encrypted format).
  Applies only to the CR 9 runtime components version of Visual CUT.

  A Note to Existing Users of Visual CUT (CR 9 Runtime Components Version)

  Since the export format options are saved in a new table (***Report_Export_Options***) in the ***Visual CUT.mdb*** database, existing users must copy that table structure into their existing ***Visual CUT.mdb*** in order to take advantage of this new functionality:

  1. Download and use MS Access to open **Visual_CUT_Export_Opt.mdb**:
  http://www.MilletSoftware.com/Download/Visual_CUT_Export_Opt_Versions.zip

  2. Use MS Access to also open **Visual CUT.mdb** and **copy** the ***Report_Export_Options*** table from **Visual_CUT_Export_Opt.mdb** into **Visual CUT.mdb**

  3. Add the ODBC export option to the **Export_Opt** table in Visual CUT.mdb

Export Constant	Export Name
CrEFTODBC	**ODBC**

Version 2.7.8000: Released 10/25/2003

- Fixed a recently introduced wrong version of export format options (**Export_Opt** table in **Visual CUT.mdb**) in the **8.5** runtime components version of Visual CUT.

  Existing users of that version of Visual CUT (**8.5** runtime components) were not impacted because, in order to preserve your settings, **Visual CUT.mdb** doesn't get changed when running REMOVE and then reinstalling newer versions.

  If you recently installed for the 1st time Visual CUT with the **8.5** runtime components version (or if you switched between the **9.0** and the **8.5** versions) you can check to see if you have the correct export format options by opening the **Export_Opt** table in **Visual CUT.mdb**. Your **Export_Opt** table should contain the older export format options for Excel (for example, **crEFTExcel50**) while the Crystal 9 runtime components version has the newer export formats (for example, **crEFTExcel97**).

  If you have the wrong version of the **Export_Opt** table:

  1. Download and use MS Access to open **Visual_CUT_Export_Opt.mdb**:
  http://146.186.176.195/CUT/Visual_CUT_Export_Opt_Versions.zip
  2. Use MS Access to also open **Visual CUT.mdb** and rename the **Export_Opt** table to **Export_Opt_90_Version**
  3. Copy the **Export_Opt_85_Version** table from **Visual_CUT_Export_Opt.mdb** into **Visual CUT.mdb** and rename it as **Export_Opt**

- Since switching between the 8.5 and the 9.0 runtime version of Visual CUT can lead to the same issue as described above (Visual CUT.mdb from the previous version remains with the wrong version of **Export_Opt** table) Visual CUT now contains two more tables (**Export_Opt_85_Version** and **Export_Opt_90_Version**) to facilitate renaming and using the correct version.

Version 2.7.7000: Released 10/20/2003

- The "From:" email option can now include a meaningful name besides the e-mail address, provided it is structured as a name enclosed in double quotes, followed by the e-mail address enclosed in "< >"
  For example: "Ido Millet" <ixm7@psu.edu>
  Thanks to Dave Clutter (Reed Manufacturing) for suggesting this enhancement.

Version 2.7.6000: Released 10/12/2003

- Added a **button for launching the user manual** from the 1st Tab of Visual CUT.

- The ini file (DataLink_Viewer.ini) now **shows all options** even if they are blank or left at their default values. This makes it **easier to identify and set these options**.

- Incorporated an **updated e-mail component** (vbSendMail version 3.65 instead of 3.54) providing the following relevant improvements:
  - Removed extra blank line from the beginning of the message body
  - Changed login authentication code to improve compatibility
  - Other minor bug fixes

- Added support for **e-mail authentication** options (User ID & Password). This change aims mainly at **avoiding the need to set the Email Server to allow *Relay***.
  Instead, the server authenticates the sender (Visual CUT). You specify e-mail authentication information by opening **DataLink Viewer.ini** (located where you installed ***Visual CUT***) and add 2 entries to the **[Options]** section:
  Email_User_ID = your_User_ID
  Email_Password = your_Password

  Thanks to Greg Davis (Act Solutions) for suggesting and testing this enhancement.

- Added support for **"POP First" authentication**. Some servers (e.g., yahoo.com) require POP3 authentication before allowing SMTP transactions. In such cases, besides providing the **Email_User_ID** and **Email_Password** ini file options (as shown above), you also need to specify an **Email_POP3_Server** address (in 123.45.678.9 format or as a domain name, such as mymailhost.com). For example:
  Email_POP3_Server = mymailhost.com

- Added support for **overriding the SMTP Port** (default is 25). If your SMTP server is not using port 25, open **DataLink Viewer.ini** (located where you installed ***Visual CUT***) and add 1 entry to the [Options] section:

Email_SMTP_Port = your_port_number

- Added a **browse button to allow easy selection of export file names.**
  Thanks to Don Gilsdorf (Gain Focus Technologies) for the suggestion.

- Fixed an issue with single-pass printing to multiple Printers.

- **Removed the size limitation (250 characters) on the total length of export file name, email to, email copy to, email blind copy to, and parameter values.**

  Since installing new versions of Visual CUT doesn't replace the ***Visual CUT.mdb*** file in existing installations, those users who desire these changes for their existing database should open their ***Visual CUT.mdb*** file (using MS Access) and change the data type of the following fields in the **Report_Opt** table from **TEXT** to **MEMO:**
  Export_File_Name, **Email_To**, **CC_To**, **BCC_To**,
  Parm1, **Parm2**, **Parm3**, **Parm4**, **Parm5**, **Parm6**, **Parm7**, **Parm8**

  Thanks to Tim Dunevant (The Matworks) for the suggestion.

- Added a new **After_Burst_Batch** command line argument for launching a batch file after each bursting cycle (and waiting for that batch file to complete processing).
  Embedded field/formula names in the batch file are temporarily replaced with dynamic values (just as other options in the 3rd tab).

  Among other uses, this allows **synchronized interweaving of group bursting printouts from multiple reports without the limitations of subreports (subreports can't use different paper trays, different paper orientations, and nested subreports).** For full detail, see the new section on: **"Interweaving Burst Printouts From Multiple Reports."**

- Added support for **specifying, in the command line, number of copies for printouts using a "Print_Copies" argument**. For example, using:
  "C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Test\Report.rpt" "Print_Copies:3"
  would override the "number of copies" option saved in Visual CUT's 3rd tab and print 3 copies of the report.

- **Added dynamic control of print copies** (whole report or each group burst).
  Imagine, for example, that your report (grouped by Order_ID) needs to print labels for each order according to how many line items the order contains. You can place a (suppressed) formula returning the count of line items in the Group header or footer. In the 3rd tab of Visual CUT, drag that formula into the print copies option (and delete the number that was there before).
  If you now invoke Print Bursting (using Printer_Burst or Printer_Burst_Only command line argument), the label for each order will be printed the appropriate number of times.

  Since installing new versions of Visual CUT doesn't replace the ***Visual CUT.mdb*** file in

existing installations, existing users who desire to take advantage of this change should open their ***Visual CUT.mdb*** file (using MS Access) and change the "**Copies**" **field size** in the **Report_Opt** table **from 3 to 50**.

Version 2.6.0000: Released 8/17/2003

- Added an option to combine multiple worksheets resulting from bursting exports to Excel into a single excel worksheet with a separate tab for each group export.
  See the "***Combining Excel Bursting Exports into a Single Multi-Tab Spreadsheet***" section in the user manual for more detail.

  Thanks to Vik Mohindra (Spryer Soft Inc.) for suggesting and testing this enhancement.

- **Changed the default excel export format from fixed to variable column widths (based on objects in the 'Whole Report'),** matching the default behavior in Crystal.
  (Note: applies only to the CR 9 runtime components version of Visual CUT).

- **Added an option allowing you to ask Visual CUT to supply yesterday or today's date as the parameter value for a discrete date parameter.**
  This allows you to use the same report interactively (specifying any date as the parameter value) as well as in scheduled mode.
  It can also lead to faster report execution since using ***currentdate*** within the report can force record selection to be performed by Crystal instead of by the DBMS.

  One way of doing this is to specify **Yesterday** or **Today** as the parameter value in a command line invocation of Visual CUT. For example:

  "C:\Program Files\Visual CUT\Visual CUT.exe" -e "C:\Test\Report.rpt" "Parm1:Today"

  Another way of doing this is by opening **Visual CUT.mdb** and entering **Yesterday** or **Today** in the appropriate parameter column within the **Report_Opt** table.
  Note that such a manual entry in the **Report_Opt** table would be overwritten if you interactively open the report in Visual CUT and click SAVE.

  Thanks to Blaise Masse **(**University of New Hampshire**)** for suggesting and testing this enhancement.

Version 2.5.3800: Released 7/15/2003

- This version **fixes a problem causing Visual CUT to ignore new parameter values when exporting a report opened interactively by the user if the report has previously saved settings with different parameter values** (and the START button is clicked without first clicking SAVE). Instead, the parameter values last saved with the report in Visual CUT were used.

  Thanks to Lori Fraticelli (K. Hovnanian Enterprises) for identifying the problem and testing the new version.

- **Minor improvements to exporting speed**.

- Minor enhancements to error messages.

- **Compiled using recently updated runtime Merge Modules**
  (applies only to the CR 9 runtime components version of Visual CUT).

Version 2.5.3700: Released 7/07/2003

- **Fixed a PDF bookmark issue** (duplication/misplacement of bookmarks) when generating the group tree of bookmarks in PDF exports.

  Thanks to Larry Bates (Syscon) for identifying the problem and testing the fix.

Version 2.5.3600: Released 6/12/2003

- Fixed an overflow problem when bursting a report with more than 326 different group values at Level 1.

  Thanks to Bill Arruda (Pragmeta) and Ben Pomeranz (WorldCom) for identifying the problem and testing the fix.

Version 2.5.3500: Released 5/25/2003

- Visual CUT can now create a "Group Tree" of bookmark inside exported pdf document. This functionality allows users to easily navigate through large PDF files.

  You specify the desired Bookmark labels (and the page they point to) by inserting one formula for each Group Header that contributes a level to the "Group Tree".

  For more information, see the section on ***Creating a "Group Tree" of Bookmarks in Exported PDF File***s in the Visual CUT User Manual.

- After PDF exports, Visual CUT now sets the summary information of the resulting pdf document according to the summary information set for the report in Crystal.
  These properties include Author, Title, Subject, and Keywords.

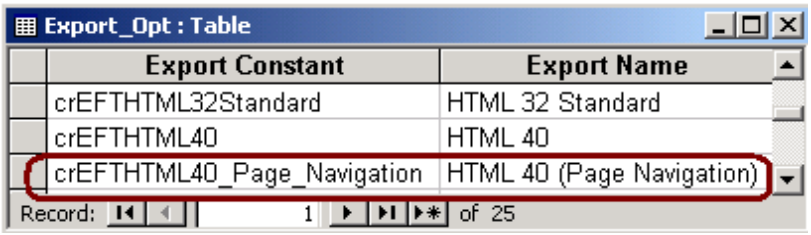
  For more information, see the section on ***Setting PDF Document Properties After Export*** in the Visual CUT User Manual.

- For export and/or e-mail bursting processes, the progress window now shows a progress bar and a counter:

  Thanks to Jon Palmbak (LeBoeuf, Lamb, Greene & MacRae, L.L.P.) for suggesting and testing these enhancements.

- Added an Export Format option of ***HTML 40 (Page Navigation)***
  This option exports each page to a separate HTML file
  (under the specified export file folder) and **adds HTML page navigation links** to the page footers.

  To enable this option, users with existing Visual CUT installations should add the following highlighted record to the ***Export_Opt*** table in ***Visual CUT.mdb***



  Thanks to Subu Swayam (Clark County Washington) for suggesting and testing this enhancement.

Version 2.5.3400: Released 4/16/2003

- The **Silent Failure option** for **Scheduled/Unattended Execution** via Command Lines or Batch Files (see Update History for Version 2.5.0) now avoids message boxes in cases where e-mail processing failed. Instead, the error message is recorded in ***Failure.log***

- A new **Silent Failure option** for **Attended Execution** (user presses the START button interactively) was added. This option allows users to hit the START button and walk away when the processing takes a long time.
  When this option is turned on, any errors (e.g., one of the Groups didn't have an e-mail address associated with it) are recorded in ***Failure.log*** instead of halting the execution with a message box.

  Here is an example of such an error message in the Failure.log:

  An attempt to send e-mail failed for the following reason(s): No Recipient E-mail Address Specified
  C:\Program Files\Visual CUT\Visual_CUT_with_Deliberate_Email_Problems.rpt: 4/9/2003 1:06:43

  You turn on this option by opening **DataLink Viewer.ini** (located where you installed Visual CUT) and specifying the following option:

[Options]
Silent_Attended_Failure=TRUE

Thanks to Jon Palmbak (LeBoeuf, Lamb, Greene & MacRae, L.L.P.) for suggesting and testing these changes.

- Fixed a problem with **redundant prompting for Parameters** with **Currency data type** (during **interactive as well as scheduled processing** of reports).

  Thanks to Helen Hiebert (Independent Consultant) for identifying the problem and testing the fixed version.

- Initial attempt to connect (before prompting user) is now made without the User ID stored by Crystal for each table. This avoids logon problems against certain data sources (e.g., SQL Server) when the data source uses integrated (NT) authentication or when the ODBC data source includes saved settings for User ID & Password..

  Thanks to Dave Neubauer (Bekaert Specialty Films) for identifying the problem and testing the fixed version.

- E-mail message text that starts with **<HTML>** and ends with **</HTML>** is now interpreted as HTML allowing fancy formatting effects.
  Thanks to Nathaniel Wetherbee (PFPC) for the suggestion.

Version 2.5.3200: Released 3/29/2003

- Burst-exporting to Crystal RPT files (with saved data) results in files as big as the original (non-bursted) file size, even though these files show only the bursted portion. **Visual CUT can now take an extra step that shrinks the size of these RPT files**.
  You turn on this option by opening **DataLink Viewer.ini** (located where you installed Visual CUT and automatically created the 1st time you run Visual CUT) and specifying the following option:
  [Options]
  Shrink_Burst_to_RPT_Files=TRUE

- Added a Command Line option to **burst a report to a printer** (to take advantage of **automatic stapling** or **faxing**).

- Fixed a problem where unattended processing of reports using Oracle Stored Procedures failed with a logon prompt (unless the scheduling string specified the user_id & password).

  Thanks to Ben Pomeranz (WorldCom) and Shawn Wright (Shawnigan Lake School) for suggesting and testing these changes.

Version 2.5.102: Released 2/27/2003

- **Removed the size limitation (250 characters) on the total length of e-mail Attachment file names.**
  Since multiple files can be attached to a single e-mail (using '**;**' to separate the file names),

one user ran into a need to specify file names that add up to more than 250 characters. The new version has no size limit on that option.

Current users should simply open their ***Visual CUT.mdb*** and change the data type of the ***Attachments*** field in the ***Report_Opt*** table from **TEXT** to **MEMO**.

Thanks to Bernard Paes (Flag Choice Hotels) for suggesting the change.

Version 2.5.101: Released 2/10/2003

- **Removed domain name checking from e-mail addresses.** This is due to a variety of new domain names being added to the internet over time.

Thanks to Larry Bodie (Kelsan, Inc.) for suggesting the change.

Version 2.5.1: Released 1/22/2003

- **Fixed an issue with batch file scheduling of reports for printout only.**

Thanks to Mike Marsten (UCSF) for identifying the issue and testing the fix.

Version 2.5.0: Released 12/17/2002

- **Reports that result in zero records can now be viewed and set up interactively.** Thanks to Abbas Rostami (Cubic Corporation) for suggesting this functionality.

- Schedule/Unattended Execution via Command Lines or Batch Files now allows silent logging of any failures and avoiding message boxes. The details about the failure get logged into a text file (Failure.Log) located where you installed Visual CUT. You turn on this option by opening **DataLink Viewer.ini** (located where you installed Visual CUT and automatically created the 1st time you run Visual CUT)
  and specifying the following option:
  [Options]
  Silent_Unattended_Failure=TRUE

- In order to respond (for example via Batch File IF THEN logic) to Success or Failure of Visual CUT processing, you can now check for the existence of **Success.txt** (in the same directory where Visual CUT is installed) after each invocation of Visual CUT processing via a command line.

  You turn on this option by opening **DataLink Viewer.ini** (located where you installed Visual CUT and automatically created the 1st time you run Visual CUT)
  and specifying the following option:
  [Options]
  Create_Success_File=TRUE

  Thanks to Mike Marsten (UCSF) for suggesting and testing this enhancement.

Version 2.4.8: Released 12/10/2002

- **Fixed a recently-introduced bug causing exporting failure** ("The remaining text does not appear to be part of the formula") **when Bursting a report that doesn't have a Record Selection Formula.**
  Thanks to Rick Scero from SITEL for catching the problem.

- Added functionality to allow **command line arguments to override the (encrypted) login information** saved for the report in Visual CUT
  (for example, … **"user_id:dba" "password:sql"**).
  In cases where different users own different schemas in the same database, this allows dynamic switching between these schemas.
  Thanks to Abbas Rostami (Cubic Corporation) for suggesting this functionality.

- Modified the **Login_Opt** table in **Visual CUT.mdb** to allow handling of cases where the Crystal runtime components return a blank **Server Name** in the connection properties of a table. Note: **if you already have Visual CUT installed you need to do the following before installing this new version**:
  1. Before installing,
     rename your **Visual CUT.mdb** file (to **Visual CUT OLD.mdb**)
     to allow the install to include the new **Visual CUT.mdb** file.
  2. Run REMOVE and INSTALL using the new msi file
  3. Open Access twice: one instance with **Visual CUT OLD.mdb**
     and the other instance with **Visual CUT.mdb**
  4. Select and copy (Ctrl-C) the records in the old **Login_Opt** table
  5. Switch to the new **Visual CUT.mdb**, open the new **Login_Opt** table
     and append (Edit, Paste Append) the records into it.
     this allows you to retain the (encrypted) login information for the reports you added to *Visual CUT*.
  6. Copy the **Report_Opt** table from **Visual CUT OLD.mdb**
     over (overwrite) the same table in **Visual CUT.mdb**
     this allows you to retain the information about the reports you added to *Visual CUT*.

- Implemented an **improvement to exporting speed**.

Version 2.4.4: Released 11/21/2002

- Added dll files for **exporting to HTML** on PCs that may be missing those files.
- The new version tolerates command line arguments that end with spaces.

Version 2.4.3: Released 11/15/2002

- Added support for exporting to and dynamically creating subdirectories using **UNC** in addition to **Mapped Network Drive** references. For example:
  \\server123\ixm7\{@Year}\Sales for {Product_Type.Name}.pdf
  Instead of
  P:\{@Year}\Sales for {Product_Type.Name}.pdf
  Thanks to Mike Marsten (UCSF) for suggesting and testing this enhancement.

Version 2.4.0: Released 11/11/2002

- **Fixed a bug causing failures when saving (interactive mode) and using (scheduled or command line mode) the encrypted logon information for reports.**

- When exporting to non-existent or non-permissible drive & directory, the user (during an interactive session) now gets a message box explaining the problem, an indication that exporting failed (in the progress window) and a clean return to the third tab.

- Fixed a problem of failed scheduled printing without command line Printer argument.

- When clicking Cancel from a log-in dialog box, Visual CUT now avoids repeated attempts (log-in windows) to log-in to each table in the main report as well as in the subreports. Thanks to Mike Marsten (UCSF) for identifying and testing most of the issues above.

Version 2.3.5: Released 11/02/2002

Added support for single-pass distribution of Crystal reports to multiple printers on the network:

- "**Printer_Only:**" argument (e.g., "**Printer_Only:\\sobpc10\sobhp04**") now allows you to schedule printing and skip any exporting, bursting, and e-mailing options saved for the report in Visual CUT. This allows you to save the report once in Visual CUT and control which functionality (printing or exporting & e-mailing) via command line arguments.

- Visual CUT can now accept an **unlimited number of "Printer" and/or "Printer_Only" command line arguments** and send the report to all these printers. This **improves performance**, compared to specifying each printer destination in a different command line, because **the report gets retrieved and processed only once**.

Version 2.3.0: Released 10/31/2002

- Added functionality to allow a **command line argument** (for example, "**Priner:\\sobpc10\sobhp04**") to **specify the Printer destination**. This allows you to save a Crystal report once within Visual CUT but schedule printouts to different printer destinations on your network.

Version 2.2.0: Released 10/30/2002

- Added functionality to allow command line arguments (for example, "Parm1:1998") to override the parameter values saved for the report in Visual CUT.

This allows external control of parameter values without the need to open the report in Visual CUT, specify new parameter values, and save the new settings each time new parameter values are needed.
Thanks to Jorge Vazquez (Merck) and Mike Marsten (University of California San Francisco) for the enhancement idea.

- Corrected the Tab sequence in the login dialog (used for secure data sources).

Version 2.1.0: Released 10/21/2002

- Fixed electronic bursting problem when grouping on text with single quotes (for example, O'Keefe). The new version can handle any number of single quotes embedded in the text of the field/formula used for Grouping at level 1).

  Thanks to Dave Hawkins from www.orbitz.com for catching the problem.

Version 2.0.0: Released 10/10/2002

- **Added compatibility with Crystal Reports Version 9.0**

- Modified Print button to allow selection of printers.

Version 1.4.0 Released 9/28/2002

- Implemented new functionality allowing users to control how empty reports (zero selected records) are handled during scheduled execution:

  Using the **–e** command line argument aborts execution of such reports silently (no message boxes that could interfere with continued processing of other scheduled repots). This is the correct choice if you have reports that should be e-mailed to users only if they contain records. This is a typical approach for exception reports.

  Using the new ***–E*** command line argument forces execution of empty reports. This is the correct choice if you have reports that should be exported and e-mailed to the users even if empty.

  Thanks to Jim Woodin (*Diamond V Mills)* for the enhancement idea.

- The Status Bar now shows the number of records **selected** in the report instead of number of records **read** from the database.

- Visual CUT can now be installed on PCs without a Crystal Reports installation.

Version 1.3.3 Released 9/07/2002

- Integrated the functionality of ***DataLink Viewer*** into ***Visual CUT***.
  This allows you to select parameter values based on live data in the database. You can experience the new functionality by simply using Visual CUT to view the following sample report: **DataLink_Viewer_Year_and_Product_Prompts.rpt report**
  For more detail, see the ***DataLink Viewer*** user manual at www.MilletSoftware.com

- Added Minimize & Resize capabilities to the Progress Window.

- Blocked the display of Group Level 1 fields & Formulas on the 3rd Tab in cases where the bursting option is disabled. Only fields from the Report Header or Footer are available in such cases. The logic is that if no bursting is possible, only values for the whole report are relevant in specifying e-mailing options.

- In cases were scheduled or unattended processing is initiated (using the **–e** command line argument) the main form as well as the progress window are now minimized as icons on the Task Bar. You can restore the progress window (by clicking the Visual CUT icon on the Task Bar) in case you want to observe or stop the unattended processing. Thanks to Jim Woodin (*Diamond V Mills)* for the enhancement idea.

- Changed the user interface (hourglass icons, status bar messages, incrementing group value count) to clarify to the user what is going on when opening of a report takes a long time. Very large reports or reports with lots of subreports take some time to a) open in the Preview Tab, and b) Process Group Level 1 values and formulas when moving to the processing (3rd) tab. Thanks to José L. Alsina (Innovatec) for the enhancement idea.

Version 1.2.5 Released 8/09/2002

- Tab 3 now shows what printer was associated with the report in Crystal (using the ***File, Printer Setup…*** dialog). Visual CUT sends reports scheduled for printing to that printer. Within Crystal, you can associate each report with a different printer using the ***File, Printer Setup…*** dialog.
  Thanks to Jim Woodin (*Diamond V Mills)* for the enhancement idea.

- Fixed an issue with the "scheduling string" on tab 3 not being refreshed.

- Stopped a warning message that appeared when scheduling a report only for printing.

- Minor enhancements to user interface, including better use of space when user increases the size of the window.

Version 1.2.4 Released 8/05/2002

- The progress window now shows a STOP button allowing you to abort the exporting/emailing process.

- Modified the software installation to rely on whatever exporting dll's are installed for Crystal Reports. This means that if you plan to use the exporting functionality of Visual CUT, your PC must have Crystal Reports installed. This change aims at removing exporting failures on some PC installations due to dll conflicts between Visual CUT and Crystal Reports.

Version 1.2.3 Released 8/02/2002

- The path (not just the name) of the exported file name can now be dynamic (drag-and-drop fields and formulas). If the specified subdirectory structure doesn't exist, it is created automatically. This allows for easy archiving of exported reports into **dynamically specified**

directory (or web site) structures. Thanks to Jim Woodin (*Diamond V Mills)* for the enhancement idea.

- If a report with **Saved Data** is opened interactively, Visual CUT now asks the user if current data from the database should be retrieved. If the user answers 'No', the Saved Data is used. This allows processing of reports without connection to a database. Thanks to David Parker of *VendorsInfo.Com* for the enhancement idea.
  Note: **Scheduled reports with saved data use the saved data.**

- Solved an Issue with processing options not being saved the first time.

Version 1.2.0: Released 7/30/2002

- Capture **report-level** and **group level-1** fields & formulas of **any data type** provided they are placed in **report** or **group level-1** headers and footers. The fields & formulas are captured even if they, or even the section as a whole, are suppressed.

- Support electronic bursting if Group Level-1 is **Text** or **Numeric** field/formula.

- **Print Scheduling** ability (including multiple **copies** and **collating** options).

- Manage reports that are not designed for bursting. Useful in cases where you simply want to schedule a report for printing or full exporting and e-mailing.

- Save and apply (for scheduling) **multi-value parameters** information (**discrete, range**, and even **mixed type** values can now be saved for later scheduling).

- Fixed HTML export issue.

- Improved speed of multiple e-mail operations.

- Improved the display of processing progress and added elapsed time information.

Version 1.1.0: Released 6/25/2002

- Added **prompting and encrypted management of Logon information** for **multiple data sources and subreports**.

- Added **Scheduling** support.

Version 1.0.0: Released 6/14/2002

Known Issues and Limitation

1. The Application stores **Logon** information in a strongly encrypted format to allow scheduled operations). While this information is NOT available during interactive use of the application, it allows scheduling (**-e** switch in RUN command line) using this stored logon information. Hence, make sure your PC is accessible only to those who have access rights to all reports managed by ***Visual CUT***.

2. Reports that use a Business Objects UNIVERSE as a data source can't be refreshed in Visual CUT. Instead, use the BO platform to schedule exports of such reports to a Crystal Report format. This provides an rpt file with saved data. You can then burst/process these reports via command line processing in Visual CUT, provided you turn on **'Use Saved Data in Scheduled Reports' in the Options dialog** or include ...**"Use_Saved_Data:True"** in the command line.

3. **Note: this restriction doesn't apply to Visual CUT 11.**
   The **License Agreement** for Crystal Reports 9 stipulates that:

   > "Report Distribution System" means any process or system or combination of processes or systems that is or are used to **automatically and/or regularly deliver**, share or distribute Reports, without providing any Access to a Server/Web-based Application: (a) to greater than fifty (50) end users directly, or (b) to a location that is accessible to greater than fifty (50) end users, who are reasonably likely to view or use the Reports. A Report Distribution System shall not include: (a) distribution of Reports in hard copy form; (b) **manual distribution on a one-time or ad hoc basis**; or (c) distribution of Client Applications created using the Royalty Free Runtime Software. For purposes of calculating the number of end users, you do not need to include any end user who otherwise has a valid license to use the Software or any of Crystal's other products (other than any product offered free of charge).

   This means that, **unless you use Visual CUT 11** (where the broadcasting license restriction no longer applies)**, you can legally distribute reports to more than 50 recipients only if that operation was initiated manually by clicking the Start button or by starting a batch file. So if you want to use Visual CUT to schedule report distribution to more than 50 recipients, use Visual CUT 11.**

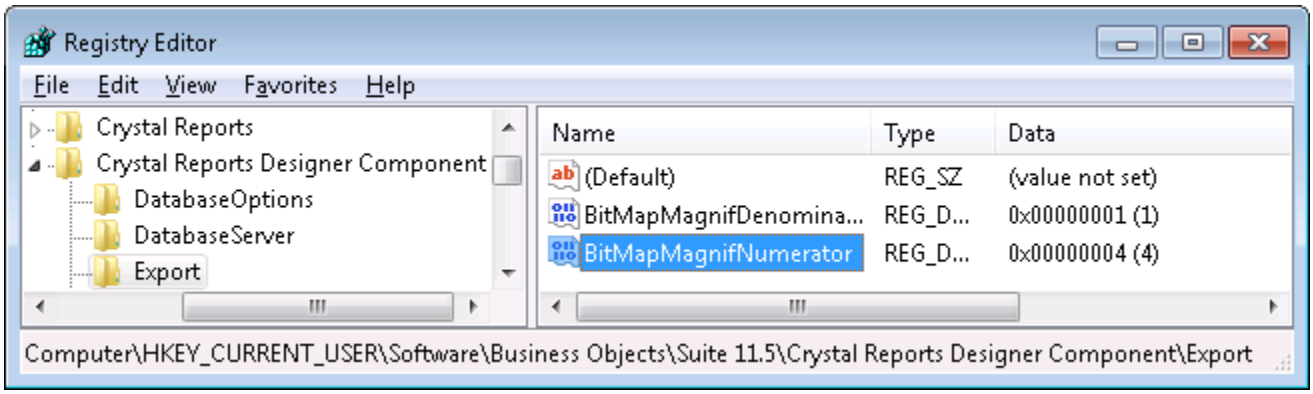
4. Patrick Pettibon from *MatchMarketing* (www.matchmarketing.com)
has identified a problem and a solution to image quality issue when previewing or exporting (PDF/RTF/Word) Crystal Reports that have charts, pictures, subreports, or OLE objects.
The problem is described by a tech support document:

You basically need to set the value of two keys in the registry:
1. BitMapMagnifDenominator (value should be 1)
2. BitMapMagnifNumerator (value should be 4)

For Crystal 8.5:
HKEY_CURRENT_USER\SOFTWARE\SEAGATE SOFTWARE\CRYSTAL REPORTS\EXPORT

For Crystal 9:
HKEY_CURRENT_USER\SOFTWARE\CRYSTAL DECISIONS\9.0\CRYSTAL REPORTS\EXPORT

However, when running a tool such as Visual CUT, similar settings are also required under a different registry folder. Here are the required settings for Visual CUT 11:



You can download a zip file with detailed instructions, Before/After images, and REG files (one each for 9.0, 8.5, and XI R2 versions) that can be imported into the registry in order to set the required keys from: http://www.milletsoftware.com/Improving_Image_Quality.zip
The usual warnings about editing the registry apply.

7. Burst-**exporting to Crystal RPT files** format results in files with saved data for the whole report because the applied filter only hides data. To ensure the resulting rpt files include only the data for each group, you can instruct Visual CUT to generate the rpt files by discarding their saved data and retrieving only the data for each group. This takes more time but **ensures that recipients can't gain access to data that doesn't belong to the exported group and that file sizes are smaller**. To use this option, turn on the "**Shrink Bursted RPT Files**" in the Options dialog. Note: **this problem doesn't affect any other export format**. This means you should not worry about this when exporting, for example, to pdf, excel, etc.

8. Exports to pdf may result in smaller font size. This is a Crystal issue, and can be fixed using some registry entries. You can do a web search for SAP document ADAPT00940675 for explanation. For more detail, see this blog by Ken Hamady:
http://kenhamady.com/cru/archives/2503#more-2503

It boils down to setting the following registry keys (for Visual CUT 11):
HKEY_LOCAL_MACHINE\SOFTWARE\Business Objects\Suite 11.5\Crystal Reports\Export\PDF\TruncationAdjustment (=2)
HKEY_LOCAL_MACHINE\SOFTWARE\Business Objects\Suite 11.5\Crystal Reports\Export\PDF\UsePrecisePositioningForText (=1)
This enlarges the font and also eliminate the truncation in most cases, and when it doesn't, you can increase the TruncationAdjustment from 2 to 3 (or higher) until the problem is resolved.

Note: On a 64-bit machine, the registry paths would start with
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\...
like this: